**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.453

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___" _____2018 р.

# Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва спеціальності)

на тему: Розподілене глибинне навчання для інтелектуального аналізу відео

Виконав (-ла): студент (-ка) __6__ курсу, групи ДА-62м
(шифр групи)

Ткаченко Дмитро Анатолійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник зав. кафедри, д.т.н., проф., Петренко А.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту д.т.н., проф., Петренко А.І. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент зав. кафедри, д.т.н., проф., Забара С.С. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

**Національний технічний університет України**
**«Київський політехнічний інститут**
**імені Ігоря Сікорського»**

Інститут/факультет ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2018 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Ткаченку Дмитру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Розподілене глибинне навчання для інтелектуального аналізу відео

науковий керівник дисертації Петренко Анатолій Іванович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження згорткові та рекурентні нейромережі для класифікації відео

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) методи машинного навчання для аналізу відео, зокрема виконання класифікації _____

5. Перелік завдань, які потрібно розробити _____
   1. Дослідження існуючих рішень, їх аналіз і пошук шляхів покращення.
   2. Розробка моделі класифікації людських дій на відео.
   3. Тестування розробленої моделі та порівняльний аналіз із існуючими.
   4. Розробка стартап-проекту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____ __
презентація на тему «Розподілене глибинне навчання для інтелектуального аналізу відео»_____

7. Орієнтовний перелік публікацій:
   1. Tkachenko D. A. Modeling spatial information with convolutional and recurrent neural networks for video classification / Dmytro Anatoliyovych Tkachenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 116-117.
   2. Tkachenko D. A. Deep learning approach to audio analysis / D. A. Tkachenko, K. V. Kharchenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 118-119.
   3. Tkachenko D. A. Human action recognition using fusion of modern deep convolutional and recurrent neural networks / Dmytro Anatoliyovych Tkachenko. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

8. Консультанти розділів дисертації*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Розробка стартап-проекту | Петренко А.І., проф. | | |
| | | | |

9. Дата видачі завдання 01.02.2018

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 01.02.2018 | |
| 2 | Огляд стану предметної області | 15.02.2018 | |
| 3 | Дослідження теоретичних основ розпізнавання відео засобами машинного навчання | 05.03.2018 | |
| 4 | Аналіз існуючих рішень | 30.03.2018 | |
| 5 | Розробка архітектури моделі класифікації відео | 10.04.2018 | |
| 6 | Реалізація запропонованого методу | 17.04.2018 | |

* Консультантом не може бути зазначено наукового керівника

| | | | |
|---|---|---|---|
| 7 | Експеримент з класифікацією людських дій на відео, порівняльний аналіз | 30.04.2018 | |
| 8 | Отримання допуску до захисту та подача роботи в ДЕК | 10.05.2018 | |

Студент _____ Д.А. Ткаченко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А.І. Петренко
(підпис) (ініціали, прізвище)

# РЕФЕРАТ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

виконаної на тему: Розподілене глибинне навчання
для інтелектуального аналізу відео
студентом: Ткаченком Дмитром Анатолійовичем

Загальний обсяг роботи: 123 сторінки, 19 ілюстрацій, 26 таблиць, перелік посилань із 57 найменувань, 1 додаток на 4 сторінках.

**Актуальність теми**

Галузь аналізу графічних даних, і зокрема відео, стрімко розвивається, і є великий попит на їх автоматичний аналіз у галузях робототехніки, безпеки, модерації користувацького контенту тощо. Ці застосування вимагають розробки моделей машинного навчання, які повинні бути точними, стійкими до шуму та цілеспрямованих атак, та дозволяти виконувати їх розподілене навчання.

Машинний аналіз відео є порівняно маловивченою галуззю через значну складність розпізнавання відео та необхідність виконання великої кількості обчислень. Однією з найважливіших задач аналізу відео є класифікація, і наразі наявно декілька підходів, які потребують дослідження та подальшого розвитку.

**Мета та задачі дослідження**

Метою даної роботи є пошук шляхів удосконалення існуючих рішень аналізу відео із застосуванням нових або маловивчених підходів. Задачею дослідження є реалізація моделі машинного навчання для класифікації відео, що досягає кращих результатів, є більш ефективною, або має інші переваги перед відомими методами.

**Вирішення поставлених завдань та досягнуті результати**

Було запропоновано архітектуру багатопотокової моделі для класифікації відео, що використовує двовимірні згорткові та рекурентні нейронні мережі, та враховує оптичні потоки і звукову доріжку. Окрім високої точності класифікації та ефективності, модель дозволяє обчислювати компактні представлення відео, що можуть застосовуватися як ознаки на вході інших моделей, для стиснення даних тощо. Як приклад застосування цих представлень було описано їх використання для виявлення аномалій.

Модель було випробувано на наборі даних, що зображає людські дії. Також було проведено експеримент із її розподіленого навчання.

**Об’єкт дослідження**

Згорткові та рекурентні нейромережі для класифікації відео.

**Предмет дослідження**

Методи машинного навчання для аналізу відео, зокрема виконання класифікації.

**Методи дослідження**

Досліджуються та застосовуються двовимірні та тривимірні згорткові, а також рекурентні нейронні мережі. Для передобробки даних і виділення ознак використовуються алгоритми обчислення оптичного потоку та частотного аналізу звукової доріжки.

Розроблене рішення використовує сучасні моделі машинного навчання та підходи до розробки їх архітектур; методи та техніки навчання, покращення точності та стійкості; а також бібліотеки для їх реалізації та розподіленого тренування.

**Наукова новизна**

Було запропоновано архітектуру моделі для класифікації відео, яка поглиблює використання ідей вивчення представлень та передавального навчання, тим самим усуваючи важливі недоліки існуючих рішень. Також було запропоновано методи вдосконалення навчання, точності та інтерпретації цієї моделі, зокрема, метод адаптивної вибірки тренувальних прикладів з урахуванням кількості інформації в сегменті відео відповідно до оптичних потоків між кадрами. Розроблена модель виконує класифікацію за вектором представлення відео, що характеризує всі потоки, за допомогою окремої моделі синтезу. Вона досягає кращих результатів класифікації, аніж відомі моделі з подібною архітектурою, і при цьому більш ефективно використовує тренувальні дані та обчислювальні ресурси.

**Практичне значення одержаних результатів**

Розроблена модель дозволяє виконувати класифікацію, а також отримувати компактні вектори представлення відео, на основі яких може реалізовуватися вирішення інших задач, у тому числі більш високорівневих, які можуть виникати при

розробці різноманітних автономних та автоматизованих методів аналізу графічних даних і керування системами.

**Апробації результатів дисертації**

Результати дослідження оприлюднені на 20-й Міжнародній науково-технічній конференції SAIT 2018, та конференції IEEE First International Conference on System Analysis & Intelligent Computing (SAIC).

**Публікації**

Tkachenko D. A. Modeling spatial information with convolutional and recurrent neural networks for video classification / Dmytro Anatoliyovych Tkachenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 116-117.

Tkachenko D. A. Deep learning approach to audio analysis / D. A. Tkachenko, K. V. Kharchenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 118-119.

Tkachenko D. A. Human action recognition using fusion of modern deep convolutional and recurrent neural networks / Dmytro Anatoliyovych Tkachenko. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

**Ключові слова**

Машинне навчання, нейронні мережі, класифікація відео, сенсорний синтез, згорткові нейронні мережі, рекурентні нейронні мережі, оптичний потік, вивчення представлень, передавальне навчання, розподілене тренування.

# РЕФЕРАТ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

выполненной на тему: Распределенное глубокое обучение для интеллектуального анализа видео

студентом: Ткаченко Дмитрием Анатольевичем

Общий объем работы: 123 страницы, 19 иллюстраций, 26 таблиц, перечень ссылок из 57 наименований, 1 приложение на 4 страницах.

**Актуальность темы**

Область анализа графических данных, и, в частности, видео, стремительно развивается, и есть большой спрос на их автоматический анализ в областях робототехники, безопасности, модерации пользовательского контента и т.д. Эти применения требуют разработки моделей машинного обучения, которые должны быть точными, устойчивыми к шуму и направленным атакам, а также позволять выполнять их распределенное обучение.

Машинный анализ видео является сравнительно мало изученной областью из-за значительной сложности распознавания видео и необходимости выполнения большого количества вычислений. Одной из важнейших задач анализа видео является классификация, и в настоящее время существует несколько подходов, которые требуют исследования и дальнейшего развития.

**Цель и задачи исследования**

Целью данной работы является поиск путей усовершенствования существующих решений анализа видео с применением новых или малоизученных подходов. Задачей исследования является реализация модели машинного обучения для классификации видео, которая достигает лучших результатов, является более эффективной, или имеет другие преимущества перед известными методами.

**Решение поставленных задач и достигнутые результаты**

Была предложена архитектура многопоточной модели для классификации видео, которая использует двумерные сверточные и рекуррентные нейронные сети, и учитывает оптические потоки и звуковую дорожку. Кроме высокой точности классификации и эффективности, модель позволяет вычислять компактные

представления видео, которые могут применяться как признаки на входе других моделей, для сжатия данных и т.д. В качестве примера использования этих представлений было описано их использование для выявления аномалий.

Модель была испробована на наборе данных, который изображает человеческие действия. Также был проведен эксперимент с ее распределенным обучением.

**Объект исследования**

Сверточные и рекуррентные нейросети для классификации видео.

**Предмет исследования**

Методы машинного обучения для анализа видео, в частности выполнения классификации.

**Методы исследования**

Исследуются и применяются двумерные и трехмерные сверточные, а также рекуррентные нейронные сети. Для предобработки данных и выделения признаков используются алгоритмы вычисления оптического потока и частотного анализа звуковой дорожки.

Разработанное решение использует современные модели машинного обучения и подходы к разработке их архитектур; методы и техники обучения, улучшения точности и устойчивости; а также библиотеки для их реализации и распределенного обучения.

**Научная новизна**

Была предложена архитектура модели для классификации видео, которая углубляет использование идей изучения представлений и передаточного обучения, тем самым устраняя важные недостатки существующих решений. Также были предложены методы улучшения обучения, точности и интерпретации этой модели, в частности, метод адаптивной выборки тренировочных примеров с учетом количества информации в сегменте видео в соответствии с оптическими потоками между кадрами. Разработанная модель выполняет классификацию по вектору представления видео, который характеризует все потоки, с помощью отдельной модели синтеза. Она достигает лучших результатов классификации, чем известные модели с подобной

архитектурой, и при этом более эффективно использует тренировочные данные и вычислительные ресурсы.

**Практическое значение полученных результатов**

Разработанная модель позволяет выполнять классификацию, а также получать компактные векторы представления видео, на основании которых может реализовываться решение других задач, в том числе более высокоуровневых, которые также могут возникать при разработке разнообразных автономных и автоматизированных методов анализа графических данных и управления системами.

**Апробация результатов диссертации**

Результаты исследования были обнародованы на 20-й Международной научно-технической конференции SAIT 2018, а также конференции IEEE First International Conference on System Analysis & Intelligent Computing (SAIC).

**Публикации**

Tkachenko D. A. Modeling spatial information with convolutional and recurrent neural networks for video classification / Dmytro Anatoliyovych Tkachenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 116-117.

Tkachenko D. A. Deep learning approach to audio analysis / D. A. Tkachenko, K. V. Kharchenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 118-119.

Tkachenko D. A. Human action recognition using fusion of modern deep convolutional and recurrent neural networks / Dmytro Anatoliyovych Tkachenko. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

**Ключевые слова**

Машинное обучение, нейронные сети, классификация видео, сенсорный синтез, сверточные нейронные сети, рекуррентные нейронные сети, оптический поток, изучение представлений, передаточное обучение, распределенное обучение.

# ABSTRACT FOR MASTER'S THESIS

on: Distributed deep learning for intelligent video analysis

by: Tkachenko Dmytro Anatoliyovych

The thesis contains 123 pages, 19 figures, 26 tables, 57 references, and 1 appendix (4 pages).

**Relevance**

The field of graphical data analysis, and specifically video analysis, is growing fast, and there is a high demand for automatic analysis tools for robotics, security, user-generated content moderation, etc. These applications require the development of machine learning models that are accurate, robust to noise and adversarial attacks, and can be trained in a distributed way.

Machine video analysis is comparatively less studied because of high complexity of video recognition and high computational demands. One of the most important problems in video analysis is classification, and currently, there are a few approaches, which require research and further development.

**Purpose**

This work aims to research ways of improving existing techniques for video analysis using new and lesser-known approaches. The goal is to implement the machine learning model for video classification that achieves better accuracy, is more efficient, or has other advantages.

**Results**

Multi-stream architecture for video classification, which uses 2D convolutional and recurrent neural networks, and takes optical flows and audio track into account, is proposed. Apart from high classification accuracy and efficiency, this model allows to extract compact video embeddings which can be used as features input to other models, for data compression, etc. Anomaly detection is described as an example of using these embedding vectors.

The proposed model has been evaluated on human action video dataset. Distributed training experiment has also been run.

**Object of research**

Convolutional and recurrent neural networks for video classification.

**Subject of research**

Machine learning methods for video analysis, specifically classification.

**Research methods**

The 2D and 3D convolutional and recurrent neural networks are studied and applied. Data preprocessing and feature extraction is done using optical flow estimation algorithms and audio frequency analysis.

The developed solution uses modern machine learning models and approaches to architecture their architecture development; methods and techniques for training, improving accuracy and robustness; libraries for implementation and distributed training.

**Scientific novelty**

The proposed architecture of video classification model utilizes ideas of representation learning and transfer learning, thus eliminating important shortcomings of existing solutions. The methods for improving training, accuracy, and interpretation of this model are proposed, particularly, the adaptive sampling method for selecting training examples based on the amount of information in a video segment according to optical flows between frames. The classification is performed based on embedding vectors, i.e., the compact representation of all streams, by a separate fusion model. The proposed model achieves better classification results than the existing models which have the similar architecture and is also more sample efficient and computationally efficient.

**Practical value**

The developed solution allows to perform classification, as well as extract compact video representation, which can be used for solving other, including more high-level, problems which arise during the development of various autonomous and automated methods of graphical data analysis and systems control.

**Approbation of research results**

Research results presented at the 20th International Conference on System Analysis and Information Technology SAIT 2018, as well as at the IEEE First International Conference on System Analysis & Intelligent Computing SAIC 2018.

**Publications**

Tkachenko D. A. Modeling spatial information with convolutional and recurrent neural networks for video classification / Dmytro Anatoliyovych Tkachenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 116-117.

Tkachenko D. A. Deep learning approach to audio analysis / D. A. Tkachenko, K. V. Kharchenko. // System analysis and information technology: 20-th International conference SAIT 2018. – 2018. – №20. – С. 118-119.

Tkachenko D. A. Human action recognition using fusion of modern deep convolutional and recurrent neural networks / Dmytro Anatoliyovych Tkachenko. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

**Keywords**

Machine learning, neural networks, video classification, sensor fusion, convolutional neural networks, recurrent neural networks, optical flow, representation learning, transfer learning, distributed training.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface, прикладний програмний інтерфейс

CNN – convolutional neural network, згорткова нейронна мережа

RNN – recurrent neural network, рекурентна нейронна мережа

LSTM – long short-term memory, довга короткочасна пам’ять

GRU – gated recurrent unit, вентильний рекурентний вузол

ReLU – rectified linear unit, усічене лінійне перетворення

SVM – support vector machine, лінійна опорно-векторна машина

SGD – stochastic gradient descent, стохастичний градієнтний спуск

GPU – graphics processing unit, графічний процесор, відеокарта

CPU – central processing unit, центральний процесор

# ВСТУП

Аналіз відео та зображень – перспективна галузь штучного інтелекту і машинного навчання, яка інтенсивно розвивається. Методи машинного навчання успішно застосовуються в багатьох  галузях. За допомогою комп'ютерного зору проводять розвідку місцевості шляхом аналізу супутникових знімків, діагностують хвороби за медичними сканами, створюють системи керування роботами та автономними автомобілями, автоматизують різноманітні процеси на виробництві (наприклад, контроль якості, тобто огляд виробів для виявлення дефектів) тощо. Але досі є багато проблем і нерозв'язаних задач, які створюють перепони для впровадження певних моделей, або характеристики таких моделей є такими, що їх використання не є економічно доцільним, швидким, точним задля вирішення конкретних задач.

Незважаючи на недоліки, аналіз зображень є дуже популярною та дослідженою сферою. Великі компанії пропонують API, за допомогою яких користувачі можуть проаналізувати свої зображення. Менш дослідженою сферою є аналіз відео. Серед задач аналізу відео можна виділити класифікацію (визначення, до якого класу або категорії відноситься відео за його змістом), тегування (пошук набору тегів, що описують зображення), виявлення послідовностей (пошук характерних послідовностей кадрів на відео, які характеризують певний процес, автоматизоване знаходження якого представляє інтерес; наприклад, виявлення певних дій), виявлення аномалій (пошук відео в наборі, які містять ознаки, які вирізняють їх від більшості інших подібних відеопослідовностей), виокремлення окремих об'єктів у кадрі, передбачення.

Вирішення кожної із зазначених задач окремо має цінність, але вони також можуть бути елементами більш складних систем прийняття рішень. Скажімо, алгоритм керування безпілотним автомобілем виконує виокремлення об'єктів на кадрах (знаків, пішоходів, інших авто), які потім класифікуються. Відеоряд також використовується для передбачення пересувань інших учасників дорожнього руху,

що є необхідним для створення автономних автомобілів, які здатні безпечно функціонувати на дорогах без спеціалізованої інфраструктури поряд із машинами, керованими людьми, та без використання протоколів комунікації «автомобіль-автомобіль» (vehicle-to-vehicle).

У цій роботі досліджуються існуючі алгоритми, методи та моделі, що застосовуються для вирішення задач аналізу відео; вивчаються та порівнюються архітектури нейронних мереж та інших моделей; аналізуються параметри цих рішень з точки зору точності класифікації, продуктивності, і визначаються переваги та недоліки існуючих рішень; досліджуються можливості паралельного та розподіленого тренування моделей машинного навчання.

**Метою роботи** є пошук шляхів удосконалення існуючих рішень із застосуванням нових або маловивчених підходів. Для цього визначаються можливі напрями поліпшення існуючих результатів; ці пропозиції реалізуються та проводиться експеримент для порівняння з опублікованими результатами.

**Кінцевою метою роботи** є створення системи розпізнавання відео, що виконує класифікацію людських дій на заданих відеопослідовностях.

# 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНОГО СТАНУ ЇЇ РОЗВИТКУ. ЗАДАЧІ ДОСЛІДЖЕННЯ

## 1.1 Загальний огляд підходів і перспективних напрямків розвитку

Традиційним підходом до аналізу відео є розбиття його на кадри, і аналіз кадрів за допомогою згорткових нейромереж. Результат класифікації може бути обчислений на основі розгляду окремих кадрів, а також можуть бути використані рекурентні нейронні мережі для моделювання послідовності представлень кадрів.

Згорткові нейронні мережі (CNN), які наразі є найбільш широко та успішно застосовуваними моделями для аналізу зображень (класифікація, виокремлення об’єктів), були винайдені більше 15 років тому [1]. Тоді було неможливо їх ефективно навчати. З появою обчислень на відносно доступних відеокартах (GPU CUDA), які прискорили обчислення на порядки, активізувалися дослідження в цій галузі, а рішення почали впроваджувати в різні галузі. Першою насправді глибокою архітектурою стала AlexNet [2], після чого було досить багато інших дуже вдалих рішень, таких як VGG [3], Inception [4] і ResNet [5].

Наразі перспективним є дослідження можливостей застосування нової архітектури CapsNet [6, 7], яка усуває основний недолік згорткових нейронних мереж – інваріантність до переносів. Це означає, що CNN працюють лише як детектори ознак та не враховують їх взаємного розташування. Це призводить до того, що, скажімо, можна розташувати основні риси обличчя на зображенні абсолютно неприродним чином, а згорткова мережа все одно класифікує картинку як обличчя.

Іншою важливою проблемою сучасних алгоритмів аналізу зображень є проблема, продемонстрована за допомогою змагальних прикладів (adversarial examples) [8]. Було показано, наприклад, що можна взяти зображення, які з 60% впевненістю правильно класифікуються моделлю, накласти на них спеціально підібраний шум так, аби картинку неможливо було відрізнити від оригіналу (людині), а модель класифікуватиме такий приклад неправильно зі 100% впевненістю. Це

концептуальна проблема, бо модель «сприймає» зображення не так, як людина, і те, що модель не стійка (robust) до таких перетворень, ставить під загрозу її використання в критичних галузях (таких, як автономне керування автомобілем). Наразі вивчаються підходи розширення тренувальних даних (data augmentation), аби навчати алгоритми правильно розрізняти зображення після їх трансформацій.

Рекурентні нейронні мережі (RNN), що використовуються для виокремлення ознак на рівні відеопослідовності, було винайдені у 1980-х, але тільки нещодавно набули широкого поширення. Традиційні RNN погано працюють із довгими послідовностями, що було вирішено в подальших архітектурах, таких як LSTM [9], GRU [10], і їх модифікаціях. Аналіз послідовностей має досить багато задач для дослідження, серед яких можна виділити специфічні проблеми навчання таких моделей, а також інтерпретація їх результатів (наприклад, за допомогою механізму уваги [11]).

Іншим перспективним напрямком є sensor fusion (сенсорний синтез), тобто використання даних із багатьох сенсорів для прийняття рішення в певній системі. Аналіз відео природним чином ділиться на декілька потоків, оскільки є аналіз на рівні кадрів і послідовностей кадрів, також є аудіопотік, який обробляється окремою моделлю [12], і можна додати інформацію, обчисливши оптичні потоки. Виходячи з цього, напрямком дослідження може бути пошук найкращої моделі синтезу результатів із цих потоків.

Через те, що сучасні моделі потребують дуже великої кількості обчислювальних ресурсів для навчання, важливим є також дослідження алгоритмів оптимізації параметрів моделі, а також підходів до розподіленого навчання. Зокрема, наявні підходи розподілу даних по вузлам в кластері, а також розподілу моделі. Також існують методи паралельного градієнтного спуску, наприклад, HOGWILD! [13].

Також останнім часом набувають поширення 3D згорткові нейромережі (наприклад, [14]). Третій вимір враховує час, тобто такі нейромережі моделюють розподілені в часі двовимірні зображення. Варто дослідити перспективність застосування таких нейромереж.

Ще одним напрямом покращення результатів класифікації відео може бути використання нових підходів до обчислення оптичного потоку, наприклад, за допомогою згорткових нейронних мереж [15].

Підхід вивчення представлень (representation learning), що дозволяє отримати компактний опис даних, який може бути використаний іншою моделлю для виконання певних задач (таких як класифікації, виявлення аномалій тощо), наразі інтенсивно розвивається та дає перспективні результати, і також буде досліджений у цій роботі.

## 1.2 Постановка задач дослідження

### 1.2.1 Завдання роботи

Таким чином, виходячи з описаного стану досліджень, сформулюємо такі завдання:

- Обрати набори відео, які будуть використовуватися для оцінки та порівняння рішень.
- Дослідити опубліковані моделі класифікації відео.
- На основі аналізу існуючих рішень виділити їх переваги та недоліки, а також шляхи покращення.
- Дослідити та запропонувати можливі шляхи покращення поточних архітектур і моделей.
- Реалізувати запропонований метод, провести дослідження його ефективності та порівняльний аналіз із існуючими рішеннями.
- Дослідити можливості розподіленого навчання моделей класифікації відео, і, зокрема, запропонованої моделі.
- Описати практичні аспекти розробки та навчання розробленої архітектури, такі як програмне та апаратне забезпечення.
- Запропонувати спосіб використання реалізованого підходу до інших задач на прикладі виявлення аномалій.

### 1.2.2 Постановка завдання тестового прикладу

Розробити модель для класифікації людських дій на відео (таких як біг, читання, розмова тощо). Модель тренуватиметься за набором даних, що містить розмічені відео, тобто використовуватиме навчання зі вчителем (supervised learning).

## 1.3 Висновки

Таким чином, наразі підходи до класифікації відео базуються на 2D і 3D згорткових і рекурентних нейромережах. Перспективними є напрямки ширшого використання ідей вивчення представлень і передавального навчання. Також існуючі рішення можуть бути вдосконалені шляхом застосування точніших методів апроксимації оптичного потоку, наприклад, таких, що використовують згорткові нейромережі.

У завдання роботи входить дослідження існуючих методів класифікації відео, виокремлення шляхів їх покращення, розробка та реалізація запропонованого методу із подальшим тестуванням класифікації на наборі відео, що зображає людські дії.

# 2 ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ВІДЕО З ВИКОРИСТАННЯМ ЗАСОБІВ МАШИННОГО НАВЧАННЯ

## 2.1 Архітектури нейронних мереж

### 2.1.1 Штучні нейронні мережі

**Штучні нейронні мережі** (artificial neural networks – ANN) [16] – це обчислювальні системи, натхнені біологічними нейронними мережами, що складають мозок тварин.

(Штучна) нейронна мережа — це мережа простих елементів, званих нейронами, які отримують вхід, змінюють свій внутрішній стан (збудження) відповідно до цього входу, і виробляють вихід, залежний від входу та збудження.

**Нейрон.** Приймає на вхід значення ознак і повертає вихідне значення відповідно до внутрішніх ваг (рис. 2.1).

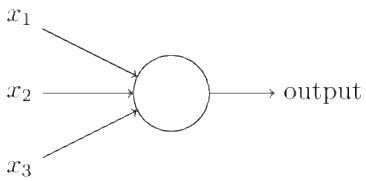



Рисунок 2.1 – Нейрон штучної нейронної мережі

Нейрон є лінійним класифікатором, його формула має вигляд лінійної комбінації ознак (або скалярного добутку) незалежних змінних і коефіцієнтів моделі:

$$\check{y}(x_1, \dots, x_i) = w_1 x_1 + \cdots + w_i x_i + b, i = \overline{1,n}, \tag{2.1}$$

де $x_i$ – значення $i$-ї ознаки;

$w_i$ – коефіцієнт $i$-ї ознаки;

$n$ – кількість ознак;

$b$ – bias-коефіцієнт, що зсуває пряму (аби вона не проходила через початок координат).

У векторному вигляді:

$$\check{y} = X \cdot W + b \tag{2.2}$$

За цим правилом нейрони комбінуються в мережу (рис. 2.2), тобто вихідні значення вхідного шару є вхідними значеннями (ознаки) на вході наступного.

Під терміном **глибинне навчання** зазвичай розуміють використання багатошарових (глибоких) нейронних мереж, що містять понад 3 прихованих шари.

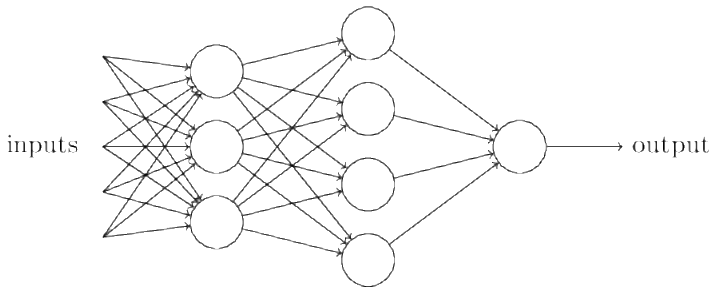



Рисунок 2.2 – Приклад нейронної мережі

**Функція активації.** Формулу нейрона можна узагальнити:

$$\check{y} = f(X \cdot W + b), \tag{2.3}$$

де $f$ – функція активації.

Таким чином, подана раніше формула нейрона відповідає лінійній $f$.

Використання нелінійної функції активації дозволяє нейронній мережі апроксимувати нелінійні функції. Згідно *теореми Цибенко* [17], нейронна мережа із як мінімум одним прихованим шаром є *універсальним апроксиматором*.

Серед активаційних функцій, що часто використовуються, можна виділити:

1. Логістична (сигмоїдальна; sigmoid). Часто застосовується в класичних методах, як то логістична регресія:

$$f(x) = \sigma(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

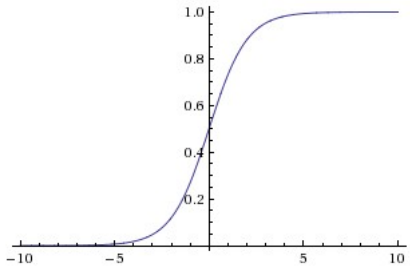



Рисунок 2.3 – Графік логістичної функції

2. Гіперболічний тангенс (tanh). Зазвичай цій функції віддають перевагу перед логістичною функцією:

$$\tanh x = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.5}$$

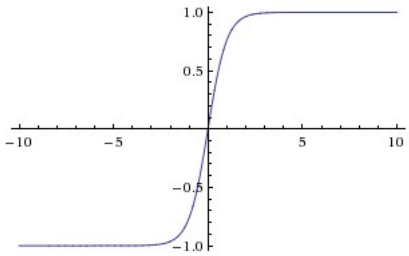



Рисунок 2.4 – Графік гіперболічного тангенсу

3. Випрямляч (ReLU, rectified linear unit – усічене лінійне перетворення). Основна перевага – простота обчислення та пришвидшення тренування глибоких мереж:

$$f(x) = x^+ = \max(0, x) \tag{2.6}$$

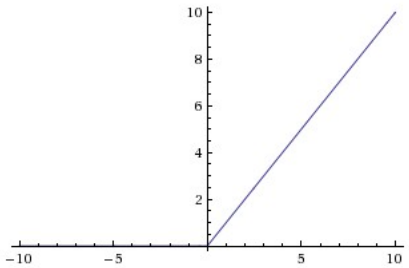



Рисунок 2.5 – Графік ReLU

4. Функція softmax є узагальненням логістичної функції та часто використовується на останньому шарі нейромереж для представлення передбачених імовірностей класів у задачах класифікації:

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{2.7}$$

**Навчання моделі.** Маючи набір даних $\{y_m, x_{m1}, \dots, x_{mn}\}_{m=1}^{k}$, що складається із $k$ прикладів – пар $(y, X)$, потрібно підібрати коефіцієнти $(W)$, аби мінімізувати певну метрику помилки (loss function, **функцію втрат**).

Скажімо, як функцію втрат можна вибрати **середньоквадратичну помилку**:

$$L = \frac{1}{k}\sum_{m=1}^{k}(y_m - \check{y}_m)^2 \quad (2.8)$$

Тобто, це середня величина квадрату різниці між істинним значенням і значенням, повернутим моделлю.

Іншою функцією втрат, що дуже часто використовується на практиці, є **перехресна ентропія (cross entropy loss)**:

$$L = -\sum_{m=1}^{k} y_m \log \check{y}_m \quad (2.9)$$

Для мінімізації функції втрат потрібно скористатися певним **алгоритмом оптимізації**. Для цього потрібно знаходити градієнт функції втрат, відповідно до значення якого певним алгоритмом оптимізації змінюються ваги нейромережі для мінімізації функції втрат. Ключовим методом його обчислення є **алгоритм зворотного поширення помилки (backpropagation)** [18].

У алгоритмі зворотного поширення помилки спочатку знаходиться прямий прохід, тобто обчислюється функція втрат відповідно до вихідного значення, яке повертає нейронна мережа для певного вхідного прикладу. Потім виконується зворотний прохід для знаходження градієнта функції втрат в кожній точці (відповідно залежно від кожного параметра мережі) за допомогою правила диференціювання складної функції та перевикористання попередніх часткових результатів.

Алгоритм зворотного поширення помилки – це ключова ідея, яка зробила можливою навчання нейронних мереж. Перевикористання попередніх результатів означає, що кількість обчислень, які необхідно виконати на кожному градієнтному кроці оптимізації ваг нейромережі лінійно залежить від глибини мережі (кількості шарів) та її ширини (кількості нейронів у шарі). Таким чином, маємо *квадратичну* складність у випадку, якщо вхідний і приховані шари мають приблизно однаковий розмір.

Найпростішим є **градієнтний спуск**. Це ітеративний алгоритм першого порядку, в якому для знаходження локального мінімуму функції здійснюються кроки, пропорційні протилежному значенню градієнту (або наближеного градієнту) функції (в нашому випадку функції втрат) в поточній точці, тобто робляться такі кроки, допоки не досягнуто певної умови зупинки:

$$w_{i+1} = w_i - \alpha \nabla L, \tag{2.10}$$

де $\alpha$ – темп навчання (learning rate).

У класичному формулюванні алгоритму градієнтного спуску градієнт обчислюється для всіх прикладів в наборі даних. Для більшості задач машинного навчання це не є можливим через великі обсяги даних, тому зазвичай використовується **стохастичний градієнтний спуск (stochastic gradient descent – SGD)**. Цей алгоритм передбачає вибір підмножини прикладів випадковим чином (міні-пакет), і обчислення градієнту лише в межах цього міні-пакету.

Існує декілька методів прискорення збіжності SGD. Серед них виділимо:

1. **Імпульс (momentum).** SGD повільно збігається в областях, де криві поверхні спадають в одному напрямі набагато швидше, ніж в іншому (так звані «яри»). Такі області часто бувають поблизу локальних оптимумів, а класичний SGD осцилює навколо схилів такого «яру» та досить повільно наближується до його нижньої точки. Імпульс прискорює SGD у відповідному напрямі та зменшує осциляції:
$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta), \theta = \theta - v_t \tag{2.11}$$
Як бачимо, це досягається за допомогою додавання частки $\gamma$ вектору попереднього часового кроку до поточного вектора оновлення ваг. На практиці, величина імпульсу зазвичай обирається рівною 0.9 або близькою до цього значення.
2. **Градієнт Нестерова (Nesterov accelerated gradient – NAG).** [19] Цей метод доповнює метод імпульсу, аби при переміщенні в бік мінімуму процедура пошуку сповільнювалася перед подальшим підвищенням рівня поверхні. Це досягається за рахунок обчислення градієнта не відносно поточних параметрів $\theta$, а відносно майбутнього значення параметрів:
$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}), \theta = \theta - v_t \tag{2.12}$$

Існує також декілька розширень градієнтного спуску, які називаються **адаптивними методами**. Такі методи є модифікацією SGD, в яких темп навчання є окремим для кожного параметра. У таких методах є базовий темп навчання, який адаптується під поведінку функції залежно від зміни кожного окремого її параметра: більші оновлення робляться для більш рідких параметрів, а менші оновлення – для

тих, які зустрічаються частіше. Це дозволяє пришвидшити збіжність у задачах із розрідженими даними. Серед таких методів часто застосовується **RMSProp (Root Mean Square Propagation)**, у якому темп навчання для параметра ділиться на рухоме середнє (moving average) величин попередніх градієнтів для цього параметра. Інший популярний метод – **Adam (Adaptive Moment Estimation)** [20] – є розширенням RMSProp і використовує другі моменти градієнтів додатково до градієнтів.

### 2.1.2 Згорткові нейронні мережі

**Згорткові нейронні мережі (convolutional neural network – CNN)** – це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень [21, 22].

Вони відомі також як **інваріантні відносно зсуву (shift invariant)** або **просторово інваріантні штучні нейронні мережі (space invariant artificial neural networks – SIANN)**, виходячи з їхньої архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення.

Архітектура CNN формується стосом різних шарів, що перетворюють ємність входу на ємність виходу (що, наприклад, зберігає рівні відношення до класів) за допомогою диференційовної функції. Зазвичай застосовується декілька різних типів шарів.

**Згортковий шар (convolutional layer)** (рис. 2.6) є основним будівельним блоком CNN. Параметри шару складаються з набору фільтрів для навчання (або ядер), які мають невеличке рецептивне поле, але простягаються на всю глибину вхідної ємності. Протягом прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідної ємності, обчислюючи скалярний добуток даних фільтру та входу, і формуючи 2-вимірну карту збудження цього фільтру. В результаті мережа навчається, які фільтри активуються, коли вона виявляє певний конкретний тип ознаки у певному просторовому положенні у вході.

Складання карт збудження всіх фільтрів уздовж виміру глибини формує повну ємність виходу згорткового шару. Таким чином, кожен запис в ємності виходу може також трактуватися як вихід нейрону, що дивиться на невеличку область у вході, та *має спільні параметри з нейронами тієї ж карти збудження*.

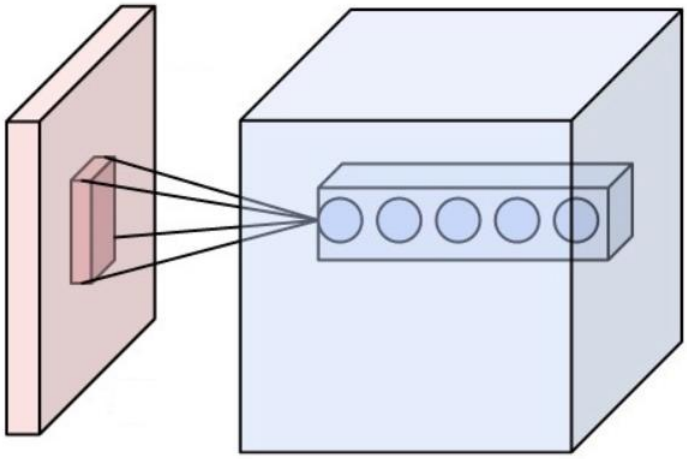


Рисунок 2.6 – Нейрони згорткового шару (праворуч), з'єднані з їхнім рецептивним полем (ліворуч)

**Агрегувальний шар** (рис. 2.7). Іншим важливим поняттям CNN є агрегування (pooling), яке є різновидом нелінійного зниження дискретизації. Існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є максимізаційне агрегування (max pooling). Воно розділяє вхідне зображення на набір прямокутників з перекриттями або без них (залежно від кроку), і для кожної такої підобласті виводить її максимум. Ідея полягає в тому, що точне положення ознаки не так важливе, як її грубе положення відносно інших ознак. Агрегувальний шар слугує для поступового скорочення просторового розміру представлення для зменшення кількості параметрів та об'єму обчислень у мережі, і відтак також для контролю перенавчання. В архітектурі CNN є звичним періодично вставляти агрегувальний шар

між послідовними згортковими шарами. Операція агрегування забезпечує ще один різновид інваріантності відносно паралельного перенесення.

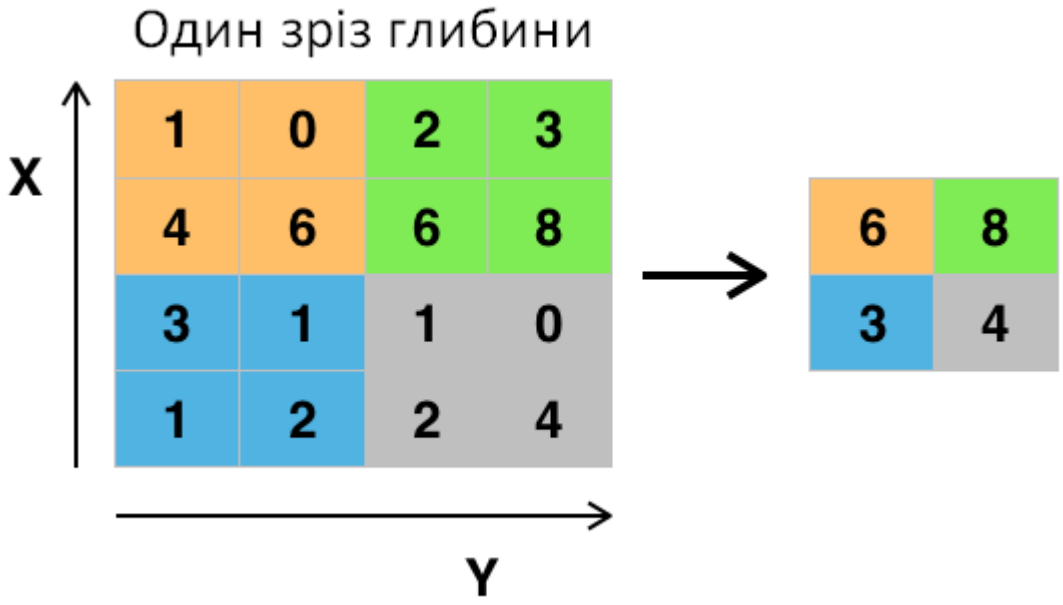



Рисунок 2.7 – Максимізаційне агрегування (max pooling)
із фільтром 2×2 та кроком = 2

**Повнозв'язний шар.** Насамкінець, після кількох згорткових та максимізаційно агрегувальних шарів, високорівневий аналіз в нейронній мережі здійснюється повнозв'язними шарами (fully connected layers). Нейрони у повнозв'язному шарі мають з'єднання з усіма збудженнями попереднього шару, як це можна бачити у звичайних нейронних мережах.

**Шар втрат.** Шар втрат визначає, як тренування штрафує відхилення між передбаченими та справжніми мітками, і є, як правило, завершальним шаром. Для різних завдань у ньому можуть використовувати різні функції втрат. Нормовані експоненційні втрати (softmax) застосовуються для передбачення єдиного класу з K взаємно виключних класів.

### 2.1.3 Рекурентні нейронні мережі

Ідея **рекурентних нейронних мереж (recurrent neural network – RNN)** – використовувати впорядковану (послідовну) інформацію [23]. У традиційній нейронній мережі вважається, що всі входи та виходи незалежні. Хоча в таких

задачах, як передбачення наступного слова в реченні, використання моделей із таким припущенням зазвичай не дає найкращих результатів. RNN називають рекурентними мережами, оскільки вони виконують однакові обчислення з кожним елементом послідовності, а вихідне значення залежить від попередніх обчислень. Також можна сказати, що RNN мають «пам'ять», яка містить інформацію про попередні обчислення. В теорії RNN можуть використовувати інформацію з послідовностей довільної довжини, але на практиці довжина послідовності обмежена. Типовий вигляд RNN показано на рис. 2.8.

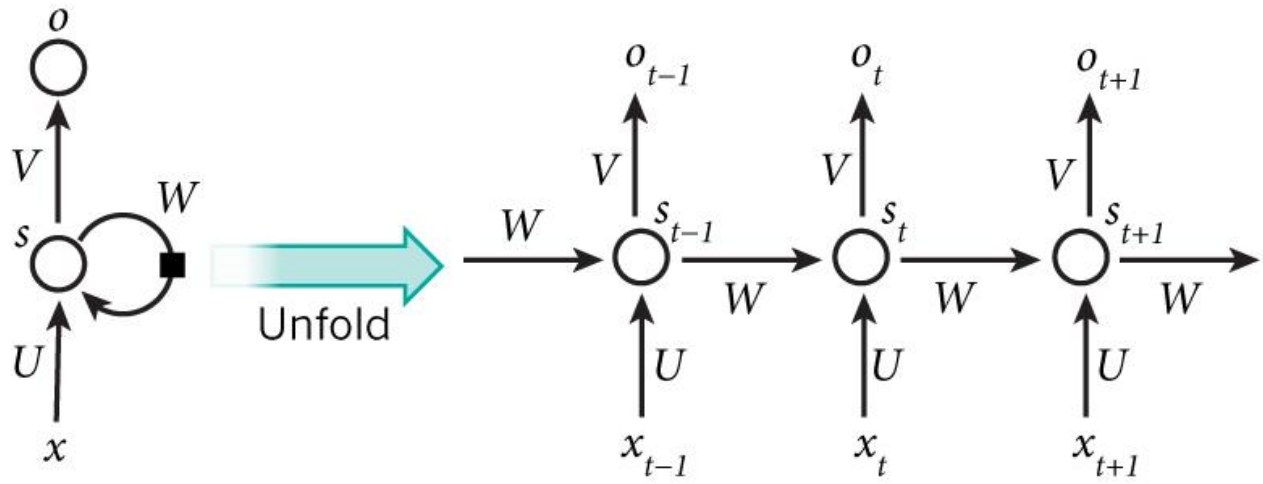



Рисунок 2.8 – Рекурентна нейронна мережа та розгортання в часі обчислень, що виконуються в прямому проході

Розгортання означає вираження архітектури мережі для повної послідовності. Скажімо, якщо ми працюємо з послідовністю з 5 слів, RNN буде розгорнуто в п'ятишарову нейронну мережу. Обчислення оперують такими змінними:

- $x_t$ – вхідне значення на кроці $t$. Наприклад, $x_1$ може бути унітарним кодом (one-hot encoding), що відповідає другому слову в послідовності (при індексації з нуля).
- $s_t$ – внутрішній (прихований) стан на кроці $t$. Це «пам'ять» мережі. $s_t$ обчислюється залежно від попереднього прихованого стану та вхідного значення на поточному кроці: $s_t = f(Ux_t + Ws_{t-1})$. Функція $f$ – деяка

нелінійність, як tanh або ReLU. Значення $s_{-1}$, необхідне для обчислення першого прихованого стану, зазвичай ініціалізується нулями.

- $o_t$ – вихідне значення на кроці $t$. Наприклад, при передбаченні наступного слова в реченні це був би вектор імовірностей слів зі словника: $o_t = \text{softmax}(Vs_t)$.

Також варто відмітити:

- На відміну від традиційної нейронної мережі, яка використовує різні параметри на кожному шарі, RNN застосовує спільні параметри ($U, V, W$) на всіх кроках. Це відображає той факт, що одні й ті ж обчислення виконуються на кожному кроці, але з різними вхідними значеннями. Це значно зменшує кількість параметрів, які потрібно оптимізувати.
- Рис. показує вихідні значення на кожному часовому кроці, але залежно від задачі це може бути непотрібним. Наприклад, для передбачення тональності речення важливе лише останнє вихідне значення, а не тональність після кожного слова. Аналогічно, не завжди потрібні вхідні значення на кожному кроці. Основна особливість RNN – прихований стан, який містить деяку інформацію про послідовність.

Навчання рекурентних мереж проводиться за допомогою **алгоритму зворотного поширення в часі (backpropagation through time – BPTT)** [24]. Центральна ідея BPTT – розгортання рекурентної нейромережі в багатошарову мережу прямого поширення для кожної послідовності, після чого алгоритм зворотного поширення застосовується для знаходження градієнту функції втрат відносно всіх параметрів мережі.

Розширення RNN:

- **Двонапрямлені RNN** базуються на ідеї, що вихідне значення в час $t$ може залежати не лише від попередніх елементів послідовності, а і від наступних. Наприклад, передбачення пропущеного слова в реченні можна робити відповідно до слів зліва та справа від нього. Архітектура проста – це дві RNN, одна з яких отримує на вхід послідовність в прямому порядку, а друга – в зворотному. Вихідні значення обчислюються відповідно до вихідних значень обох мереж (рис. 2.9).

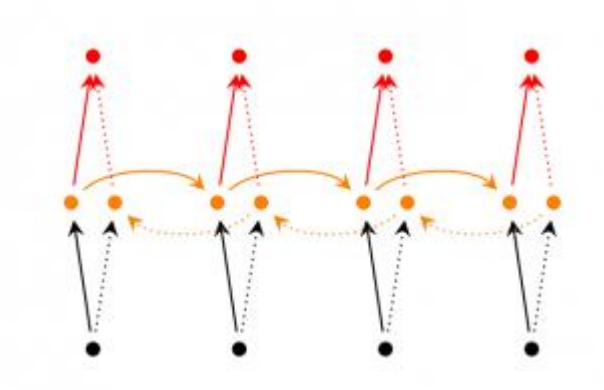


Рисунок 2.9 – Двонапрямлена RNN

- **Глибокі RNN** (рис. 2.10) мають декілька шарів на часовий крок. На практиці це дає мережі здатність вивчати більш складні залежності (але такі мережі потребують також більшої кількості тренувальних даних).

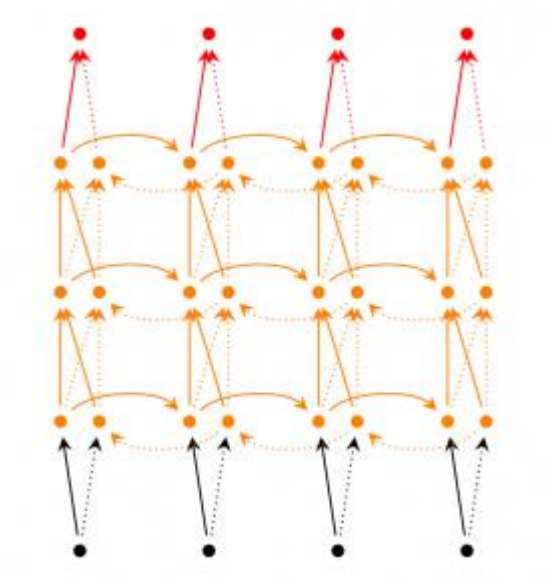
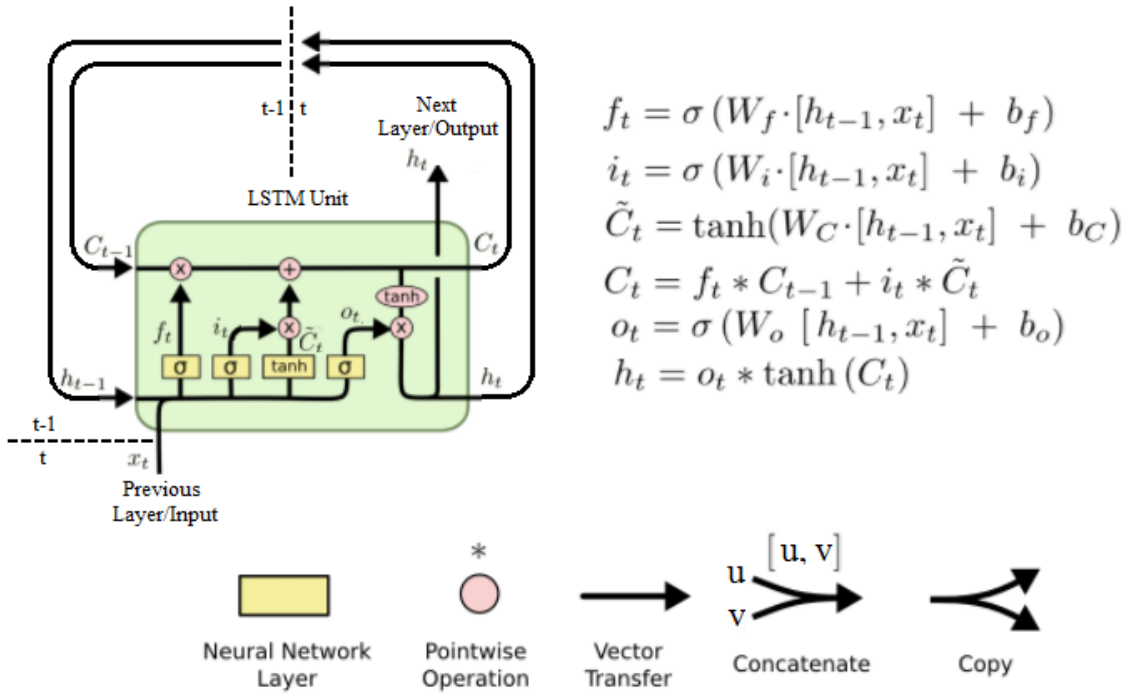


Рисунок 2.10 – Глибока двонапрямлена RNN

На практиці основною проблемою звичайних RNN є складність моделювання довгострокових залежностей. Для вирішення цієї проблеми було розроблено архітектуру **LSTM (long short-term memory; довга короткочасна пам'ять)** [9]. Вона містить спеціальні комірки RNN, що мають декілька додаткових «вентилів». Ці вентилі визначають, чи передається інформація через вентиль і якою мірою вона впливає на результат; це визначається функцією вентиля та її параметрами. Комірка LSTM складається із таких компонентів [25]:

- *Стан комірки* – з'єднання, схожого на конвеєрну стрічку, що відповідає за передачу інформації між комірками з мінімальною кількістю змін.
- *Вентиль забуття* – контролює, яку частина інформації з попередньої комірки вплине на стан поточної.
- *Вхідний вентиль* – визначає вплив вхідного значення на значення в стані комірки.
- *Вихідний вентиль* – визначає «відфільтровану» версію стану комірки, яка буде вихідним значення цієї LSTM комірки.

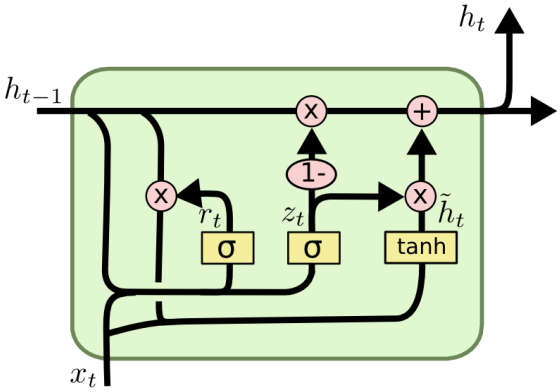
Вигляд комірки LSTM, а також формули вентилів показано на рис. 2.11.




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Рисунок 2.11 – Діаграма комірки LSTM із умовними позначеннями та формулами вентилів

Однією з часто використовуваних варіацій LSTM є **GRU (Gated Recurrent Unit)** [10]. У ній з-поміж іншого вентиль забуття та вхідний вентиль комбінуються в один (вентиль оновлення), а також об’єднуються стан комірки та прихований стан. Ця модель простіша за LSTM і має менше параметрів. Діаграму GRU із формулами вентилів показано на рис. 2.12.




$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Рисунок 2.12 – Діаграма комірки GRU із формулами вентилів

## 2.2 Оптичний потік

Одним із ключових підходів до аналізу відео є розгляд оптичного потоку. Оптичний потік може виступати як додаткова інформація, тобто характеристика даних при тренуванні моделі машинного навчання.

**Оптичний потік** — це представлення видимого сліду руху об'єктів, поверхонь, і граней візуальної сцени, що спостерігається під час відносного руху між спостерігачем (наприклад, око людини або камера) і сценою.

Послідовність з впорядкованих зображень дозволяє оцінити рух, або у вигляді миттєвих швидкостей зображення, або у вигляді дискретних величин зміщення зображення.

Методи розрахунку оптичного руху намагаються оцінити рух між двома кадрами зображення, які відносяться до моментів часу $t$ і $t + \Delta t$ в кожній позиції вокселя.

Іншими словами, оптичний потік між двома 2D зображення, які відповідають моментам часу $t$ та $t + 1$, представлений набором векторів із координатами $(u_i, v_i)$ для певної множини точок на зображеннях. Кожен із них є *вектором переміщення*,

тобто таким, що з'єднує дві точки, і довжина якого дорівнює найкоротшій відстані між початковою та кінцевою позиціями точки. Таким чином, знаходження вектору оптичного потоку полягає у визначенні вектора переміщення $(u_i, v_i)$, що з'єднує позицію певної точки $(x_t, y_t)$ в початковий момент часу $t$ та позицію цієї ж точки $(x_{t+1}, y_{t+1})$ в наступний момент часу $t + 1$.

Як точки, для яких обчислюватиметься оптичний потік, можуть бути обрані певні точки, які найкращим чином характеризують зображене (наприклад, краї об'єктів). Із розвитком апаратного забезпечення та появою нових швидших алгоритмів апроксимації оптичного потоку, стало можливим обчислювати його для всіх точок; такий підхід має назву *щільного (dense)* оптичного потоку.

Класичним методом обчислення оптичного потоку є **алгоритм Лукаса-Канаде** [26]. Основне рівняння оптичного потоку містить дві незалежні змінні і не може бути однозначно розв'язаним. Алгоритм Лукаса-Канаде усуває неоднозначність за рахунок використання інформації про сусідні пікселі в кожній точці. Метод ґрунтується на припущенні, що в локальному околі кожного пікселя значення оптичного потоку однакове, таким чином можна записати рівняння оптичного потоку для всіх пікселів в околі і розв'язати систему рівнянь методом найменших квадратів.

## 2.3 Передобробка звукового потоку

Теоретично моделі машинного навчання можуть працювати із необробленим звуковим сигналом, але на практиці можна значно прискорити та покращити їх навчання за допомогою передобробки звукового потоку та формування додаткових характеристик.

Основним методом передобробки є **перетворення Фур'є**, яке трансформує у часовому домені (залежний від часу сигнал) у сигнал у частотному домені. Часовий домен представляє сигнал як послідовність вимірювань, в той час як частотний домен подає сигнал як суперпозицію синусоїд різних магнітуд, частот і фазових зсувів. Дискретне перетворення Фур'є послідовності $\{x(n)\} = x_0, x_1, \dots, x_{N-1}$ визначається формулою:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega n}, k = 0, 1, \dots, N-1, \quad (2.13)$$

де $\omega$ – частота;

$x(n)$ – сигнал.

Обчислення перетворення Фур'є за такою формулою має складність $O(n^2)$. На практиці використовуються алгоритми швидкого перетворення Фур'є, які значно прискорюють обчислення, вимагаючи $O(n \log n)$ операцій. Також в присутності помилки округлення алгоритми швидкого перетворення Фур'є дають точніший результат порівняно з прямим обчисленням за визначенням.

Прикладом є **алгоритм Кулі-Тьюкі (Cooley-Tukey)** [27], який застосовує рекурсивний поділ вхідної послідовності значень, обчислює перетворення Фур'є для кожної частини, і поєднує результати.

Звукові сигнали зазвичай *нестаціонарні* (тобто розподіл залежить від часу), тому перетворення Фур'є для цілого аудіо треку (звукової доріжки відео, пісні тощо) призвело би до втрати інформації. Тому на практиці використовується **віконне перетворення Фур'є**:

$$X(m, \omega) = \sum_n x(n)w(n-m)e^{-j\omega n}, \quad (2.14)$$

де $m$ – час;

$\omega$ – частота;

$w$ – віконна функція (наприклад, вікно Гаусса або Гана).

Відповідно, зі збільшенням $m$ вікно зсувається вправо, а перетворення Фур'єр виконується для $x(n)w(n-m)$.

Як характеристику аудіо треку часто використовують **спектрограму**, тобто залежність амплітуди від частоти і часу, яка обчислюється як квадрат модуля віконного перетворення Фур'є:

$$S(m, \omega) = |X(m, \omega)|^2 \quad (2.15)$$

Також на практиці часто використовується масштабування спектрограми. Існує **шкала мел** – шкала висот звуку, в якій люди сприймають їх як рівновіддалені. Перетворення частот у Гц ($f$) у шкалу мел визначається формулою:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \quad (2.16)$$

## 2.4 Висновки

Було проведено теоретичний аналіз методів і алгоритмів, які використовуються для побудови системи аналізу відео. Усі моделі машинного навчання, що застосовуються, базуються на нейронних мережах. Було досліджено загальний підхід до проектування та навчання нейромереж.

Також було вивчено підходи до передобробки відео та аудіо потоків. Оптичний потік дозволяє явним чином визначити часові залежності між кадрами, що складають відео, тобто описати рух об’єктів. Передобробка аудіо, заснована на перетворенні Фур’є, дозволяє отримати спектрограму, яке характеризує аудіо трек (або його частину), і може використовуватися як додаткова ознака тренувальних прикладів (відео) при навчанні моделі.

Згорткові нейронні мережі застосовуються для аналізу окремих кадрів (або оптичних потоків), але також можуть бути узагальнені для врахування часового виміру. Рекурентні нейромережі явним чином моделюють послідовності, тому їх доцільно використовувати для послідовностей представлень кадрів або оптичних потоків.

# 3 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Згорткові нейронні мережі є найбільш успішним інструментом класифікації зображень. Як їх можна було б використати для класифікації відео? Можливо виділити окремі кадри відео та класифікувати кожен з них окремо. Для класифікації відео наївним підходом є класифікація кожного кадру окремо та агрегація цих результатів. CNN, натренована для класифікації, повертатиме вектор, в якому міститеться ймовірність належності зображення до кожного з класів. Далі можна усереднити ймовірності по всіх кадрах, і наприкінці видати клас із найбільшою ймовірністю як результат класифікації відео. Узагалі, цей підхід може видавати певні правильні передбачення, оскільки певні дії можуть асоціюватись із присутністю конкретних об’єктів на кадрі. Але такий підхід має багато недоліків:

1. Не враховується контекст. Виконується передбачення для кожного кадру окремо без урахування всіх інших, які містяться на відео. Відео – це цілісна послідовність кадрів, які разом зображають певну дію або процес. До того ж, можлива така ситуація: відеоролик зображає проїжджаючу машину, але на деяких кадрах вона не зображена. Це вплине на результат класифікації, бо, якщо в переліку класів є такий, який пов’язаний з машинами, а згорткова нейромережа асоціює наявність машини на зображенні з ним, то для тих кадрів, де машина відсутня, CNN видаватиме неправильні передбачення, що вплине на кінцевий результат.
2. Не враховується послідовність. Певну складну дію або процес на відео можна ідентифікувати за послідовністю кадрів, що розглядаються в правильному порядку, але, якщо розглядати кожен окремо або в довільному порядку, в загальному випадку такий підхід не працюватиме. Наприклад, без явного врахування послідовності, нейромережа буде не в змозі розрізнити, закриває чи відкриває двері людина на відео.
3. Не враховуються часові залежності та рух. Переглядаючи відео, люди сприймають рухомі об’єкти, будуючи ментальну модель чогось на кшталт

оптичного потоку, тобто розуміючи переміщення об’єкту в просторі [28]. Також мають значення часові параметри, тобто з якою швидкістю переміщується об’єкт.

4. Не враховується звук. Сприйняття звуку є важливою рисою сприйняття навколишньої дійсності, і врахування звукової доріжки на відео може допомогти точніше класифікувати це відео. Хоча, звісно, це залежить від запису: наприклад, навколишній шум може заглушати звуки, які характеризують дію, релевантну для класифікації. Можна знайти й інший приклад, коли сприйняття звуку є обов’язковою умовою правильної класифікації: скажімо, на відео зображений музикант, що грає на певному інструменті; але через значну відстань до спостерігача неможливо точно ідентифікувати, на чому саме він грає; при цьому може бути чутно його гру, що дасть змогу визначити, за допомогою якого інструменту він продукує музику.

Отож можна зробити висновок, що в загальному випадку застосування згорткової нейромережі для класифікації окремих кадрів і агрегація результатів не даватиме оптимальних результатів. Розглянемо, які підходи пропонуються в опублікованих працях.

## 3.1 Багатопотокові архітектури

У [29] пропонується одна з перших багатопотокових архітектур. Окрім класифікації кожного кадру окремо, автори також пропонують використовувати оптичний потік. Цей метод передбачає обчислення оптичного потоку для пар кадрів відео, та об’єднання потоків декількох кадрів в один тензор, який і подається на вхід згорткової нейромережі, що виконує класифікацію.

Опишемо запропоновану схему об’єднання. Щільний оптичний потік – набір векторів переміщення, який позначимо $d_t$ для пари кадрів у момент часу $t$ і $t+1$. $d_t(u, v)$ позначає вектор переміщення в точці $(u, v)$ кадру $t$, який переміщує точку до відповідної точки на наступному кадрі $t+1$. Горизонтальні та вертикальні компоненти вектора, $d_t^x$ і $d_t^y$, можна вважати каналами зображення, які можна подати

на вхід CNN для розпізнавання. Для представлення руху на послідовності кадрів, канали потоку $d_t^{x,y}$ об'єднуються для $L$ послідовних кадрів, формуючи $2L$ вхідних каналів. Якщо позначимо ширину та висоту відео як $w$ і $h$ відповідно, то вхідний об'єм згорткової мережі $I_\tau \in \mathbb{R}^{w \times h \times 2L}$ для довільного кадру $\tau$ конструюється так:

$$I_\tau(u, v, 2k-1) = d^x_{\tau+k-1}(u, v),$$
$$I_\tau(u, v, 2k) = d^y_{\tau+k-1}(u, v),$$
$$u = [1; w], v = [1; h], k = [1; L] \quad (3.1)$$

Для довільної точки $(u, v)$ канали $I_\tau(u, v, c), c = [1; 2L]$ кодують рух цієї точки в послідовності з $L$ кадрів. В експериментах найкращий результат показало використання довжини групи оптичних потоків $L = 10$, тобто сконкатенованих оптичних потоків між кадрами в послідовності з 11 кадрів.

Таким чином, у цій архітектурі одна CNN класифікує окремі кадри (*просторовий (spatial) потік*), а інша – сконкатеновані оптичні потоки для послідовностей кадрів (*часовий (temporal) потік*). Обидві нейромережі повертають імовірності належності об'єкту, що класифікується, до наявних класів. Автори дослідили два підходи до синтезу результатів цих двох мереж: усереднення та тренування лінійної опорно-векторної машини (support vector machine – SVM) на сконкатенованих softmax передбаченнях. SVM дала кращий результат. Загалом, цим методом було отримано точність 88% на наборі даних UCF101 [30].

Також варто зазначити, що в цій роботі використовувалося **передавальне навчання (transfer learning)**. Це – практика навчання моделі на певному наборі даних, а потім використання її для іншої задачі, тобто використання «знань», отриманих при вирішенні однієї задачі, для іншої. Наприклад, в цьому методі просторова CNN була навчена для класифікації на наборі даних ImageNet [31]. Це дуже великий набір даних для класифікації зображень (з-поміж іншого), тому навчання CNN на ньому дозволяє мережі значною мірою вивчити різноманітні геометричні патерни, притаманні фотографіям реальних об'єктів. У UCF101 різноманітність класів набагато менша (їх 101), тому використання параметрів моделі, натренованої на ImageNet, значно покращує результат. Передавальне

навчання реалізується так: параметри останнього (верхнього) шару нейромережі із вагами, отриманими в результаті навчання на ImageNet, оптимізуються для мінімізації значення функції втрат на цільовому наборі даних (UCF101). Таким чином, усі згорткові шари залишаються незмінними, модифікується лише шар, що ставить у відповідність знайдені на зображеннях характеристики потрібним класам.

При використанні передавального навчання може бути потрібним робити **тюнінг (fine-tuning)** моделі під задачу. Зазвичай тюнінг виконується для певної кількості верхніх шарів. Нижні шари мережі ідентифікують низькорівневі характеристики (базові геометричні форми – лінії, дуги тощо). Наступні шари об’єднують низькорівневі патерни та детектують більш складні характеристики (наприклад, комбінація базових геометричних фігур може об’єднуватись у обличчя людини). В результаті, параметри нижніх шарів мережі можна перевикористати для майже будь-якої задачі розпізнавання зображень, а верхні шари іноді є сенс оптимізувати під конкретний набір даних. Кількість верхніх шарів, тюнінг яких виконується, залежить від того, наскільки базовий набір даних (на якому навчалася модель) відрізняється від того, який необхідно моделювати. Наприклад, якщо модель була натренована на фотографіях, то її можна застосувати для класифікації іншого набору фотографій із мінімальним тюнінгом; якщо ж модель була натренована на фотографіях, а потрібно класифікувати медичні знімки (рентгеновські тощо), то цільовий набір даних сильно відрізняється від базового (зображення зовсім не схожі), і потрібно в процесі навчання варіювати набагато більшу кількість верхніх шарів мережі, тому що перевикористати можна лише фільтри згорткових шарів, які детектують найбільш базові геометричні патерни.

Описана у [29] модель є значним кроком уперед, тому що використання оптичного потоку послідовності кадрів дозволяє моделювати рух об’єктів. Але варто відмітити такі недоліки:

- Оптичний потік значно відрізняється від самих зображень. Він показує лише рух об’єктів, але не враховує їх вигляд. Тому оптимальним було б у просторовому потоці використовувати модель, яка враховувала би послідовність зображень або їх представлень.

- Оптичний потік на послідовності з 10 кадрів може охопити певну дію, але цього може бути недостатньо. Наприклад, у відео довжиною 30 секунд є 750 кадрів (25 кадрів на секунду). Це означає, що окремо класифікуються лише невеликі шматки (довжиною, відповідно, 0.4 секунди). Результати класифікації цих окремих сегментів об'єднуються без урахування їх послідовності. Кращий результат можна було б отримати, використавши модель, що працює із послідовністю представлень сегментів відео, яка охоплює відео повністю.

Представлений у [32] метод усуває більшість із цих недоліків. Перш за все, у ньому використовується підхід **вивчення представлень (representation learning)**. Це одна з провідних ідей у машинному навчання, яка набуває все більшого поширення останніми роками. Концепт полягає в тому, аби модель вивчала *представлення* даних, необхідні для класифікації або вирішення інших задач.

Покажемо це на прикладі розпізнавання зображень. Згорткова нейронна мережа містить набір згорткових і агрегувальних шарів. У кінці мережі є один або декілька повнозв'язних шарів, останній із яких безпосередньо виконує класифікацію (тобто вихід цього шару має розмірність, що відповідає кількості класів, і повертає ймовірності належності прикладу (зображення) до певного класу). Якщо видалити останній класифікаційний шар або «зняти» вихідні значення з передостаннього шару, цей вектор буде внутрішнім представленням зображення, яке використовується моделлю для класифікації. Наприклад, це внутрішнє представлення може мати розмірність 1024, у той час як вихідний вектор має розмірність 101 (відповідно до кількості класів). Це внутрішнє представлення називається **вектором представлення (embedding vector)**. Можна сказати, що нейромережа виконує стиснення даних; скажімо для кольорового (RGB; тобто із 3 каналами) зображення розміром 320x240, розмірність «сирого» зображення дорівнює 230400 (320x240x3). При цьому вектора представлення розмірністю 1024 достатньо для класифікації! Цей підхід схожий на підхід **word2vec** [33], що використовується в обробці природньої мови (natural language processing – NLP). Методи з цієї групи вивчають представлення слів (у вигляді векторів). У результаті у просторі векторів представлень вектори, що відповідають семантично близьким словам, знаходяться на меншій відстані

(відповідно до певної метрики відстані; наприклад, косинусоїдальна відстань (cosine distance або cosine similarity), з якою два вектори вважаються близькими, якщо кут між ними малий), ніж слова, що більше відрізняються за значенням. Загалом, це дуже зручне представлення, оскільки над векторами в просторі векторів представлень можна виконувати різні операції, наприклад, кластеризацію (формувати кластери з близьких векторів) тощо. Таким чином, отримавши embedding vector, можна використовувати його замість «сирих» даних як вхідне значення іншої моделі. При цьому цей вектор має значно меншу розмірність і містить важливі характеристики даних.

Отож запропонована у [32] модель використовує вектори представлень зображень, які продукує згорткова нейромережа. Ці вектори представлень подаються у LSTM, яка, відповідно, *класифікує відео за послідовністю представлень зображень*. Такий підхід застосовується і в просторовому потоці, і в часовому. Для сконкатенованих оптичних потоків (для послідовності кадрів із сегменту відео) інша CNN повертає вектор представлення, який також обробляється окремою LSTM.

Як вже було зазначено, використання оптичного потоку «допомагає» навчати мережу, оскільки це готове представлення додаткової корисної інформації. Але є ще одна перевага: сконкатеновані оптичні потоки характеризують низькорівневий рух із високою «розподільчою здатністю», тобто із вищою частотою вибірки (sampling rate) ніж та, що застосовується для формування послідовності, яка подається на вхід просторової LSTM. Збільшення довжини вхідної послідовності LSTM сповільнює її навчання та може викликати проблеми на невеликих наборах даних, тому частота вибірки зазвичай обирається нижчою за частоту кадрів на відео (часто 25 кадрів на секунду).

Також цей метод використовує аудіо доріжку. Спектрограма подається на вхід окремої CNN, яка виконує класифікацію.

Таким чином, маємо 5 потоків: окремі зображення ($CNN_1$), послідовність представлень зображень ($LSTM_1$ із embedding з $CNN_1$), об’єднані оптичні потоки для сегментів відео ($CNN_2$), послідовність представлень сконкатенованих оптичних потоків ($LSTM_2$ із embedding з $CNN_2$), і спектрограма ($CNN_3$). Результати

передбачення (тобто вектори із імовірностями належності відео до певного класу) цих потоків об’єднуються в фінальний результат.

Загалом [32] представляє цілісний підхід до класифікації відео, який ураховує всю доступну інформацію. *На мою думку, цей підхід доцільно розвивати надалі.* За допомогою цього підходу авторами було отримано точність 92.6% на наборі даних UCF101. Єдиним недоліком є те, що синтез потоків для отримання результату класифікації проводиться з *передбаченнями* для окремих потоків; таким чином, по суті *класифікація окремих потоків є незалежною*.

## 3.2. Згорткові 3D мережі

Згорткові нейромережі, що використовуються для класифікації зображень, оперують *2D згортками*. Ці згортки мають фільтри, що містять $F \times F \times D$ параметрів (де $F$ – розмірність (сторона; spatial extent) фільтра, а $D$ – кількість каналів вхідного об’єму (= 3 для кольорових RGB зображень)). Такі фільтри розміром $F \times F$ «сканують» вхідне 2D зображення, а результуючі активації є сумою поелементних добутків параметрів фільтру на значення у вхідному тензорі (тобто додавання також відбувається для $D$ вхідних каналів). У результаті маємо, що вихідний об’єм кожного фільтру є двовимірним (тому це 2D згортки). Таким чином, фактично ми маємо 3D вхідний об’єм (ширина, висота, кількість каналів), але на виході отримуємо 2D за рахунок додавання активацій усіх каналів. По суті для кожного каналу вводиться окремий набір параметрів, а наявність декількох каналів зображення просто збільшує розмірність; при цьому це не впливає на розмірність вихідного об’єму; інакше можна сказати, що на вході є 2D об’єм, але для кожної точки в двовимірному просторі є декілька значень (каналів); тобто глибину вхідного об’єму можна вважати рівною 1, оскільки канали обробляються окремим чином та не впливають на розмірність вихідного об’єму. Схематично це показано на рис. 3.1.

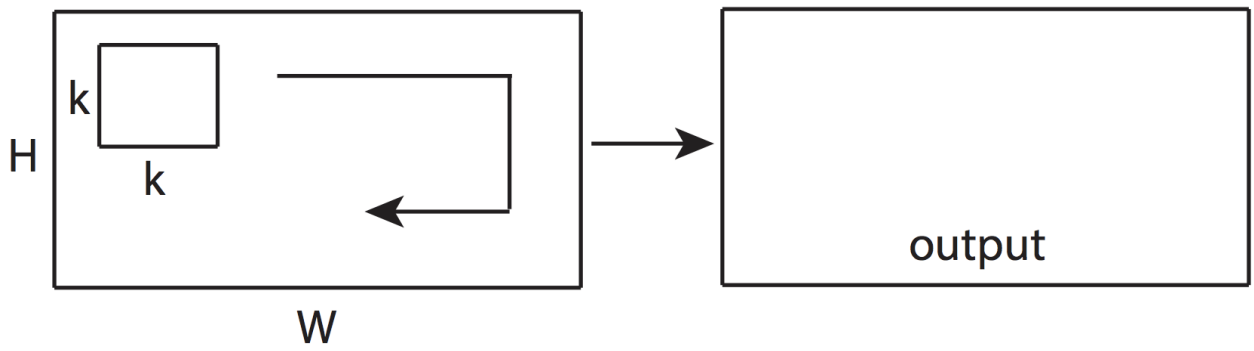



Рисунок 3.1 – 2D згортка

Як вже було зазначено, проблемою використання CNN для класифікації відео є те, що розглядаючи лише окремі кадри, втрачається контекст і порядок в послідовності кадрів. Використання сконкатенованих оптичних потоків (оптичні потоки для сегменту відео з N послідовних кадрів, об'єднані в один тензор) частково вирішує цю проблему, дозволяючи моделювати певну дію чи процес відповідно до руху в цьому сегменті. При такому підході оптичні потоки декількох кадрів в сегменті є *окремими каналами* вхідного об'єму. Цю ідею можна застосувати і для зображень: можна аналогічним чином вважати декілька послідовних кадрів у сегменті відео окремими каналами на вході CNN. Це показано на рис. 3.2.

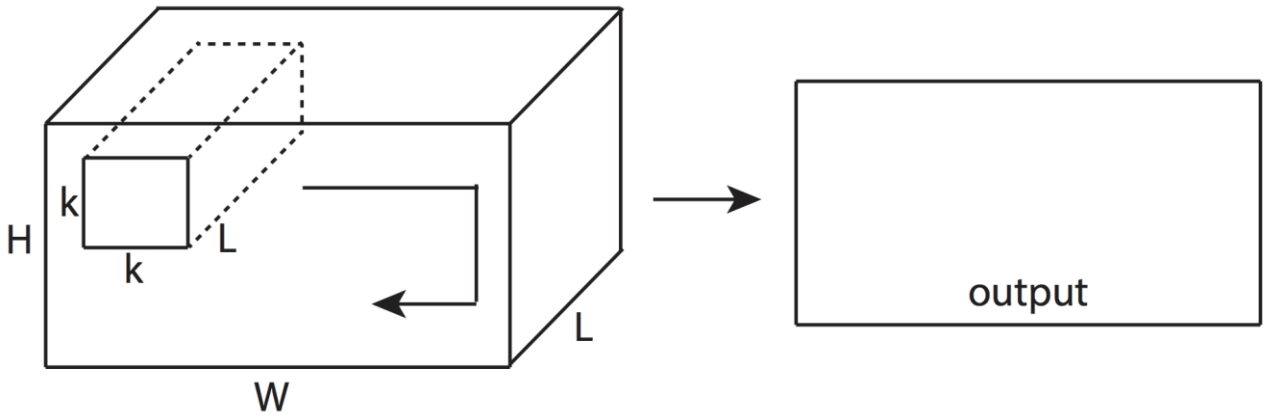



Рисунок 3.2 – 2D згортка декількох кадрів

Проблема використання 2D згорток полягає в тому, що вони моделюють лише просторові залежності. Це є результатом того, що хоча 2D згортка й може отримувати часові залежності на вхід (як окремі канали), ця *інформація втрачається на виході*. Успіх CNN обумовлений тим, що це глибокі мережі, у яких послідовні згортки моделюють дані на все більших рівнях абстракції з поглибленням мережі (початкові шари захоплюють базові геометричні патерни, подальші шари комбінують їх в більш складні і так далі). Але використання 2D згорток для класифікації відео призводить до *втрати часових залежностей після першого ж згорткового шару*. Це робить неможливим моделювання складних процесів, зображених на відео, лише за допомогою згорткових 2D нейромереж.

Логічним узагальненням є **3D згортки**. Входом є 3D об'єм, наприклад, (ширина, висота, час). Тобто, наприклад, декілька 2D зображень (кадрів). Відмінність від 2D згортки полягає у тому, що фільтр є тривимірним ($F \times F \times L$, де $L$ – глибина фільтру ($1 < L < D$)). Таким чином, *вихідний об'єм кожного фільтру є тривимірним*. Це показано на рис. 3.3.

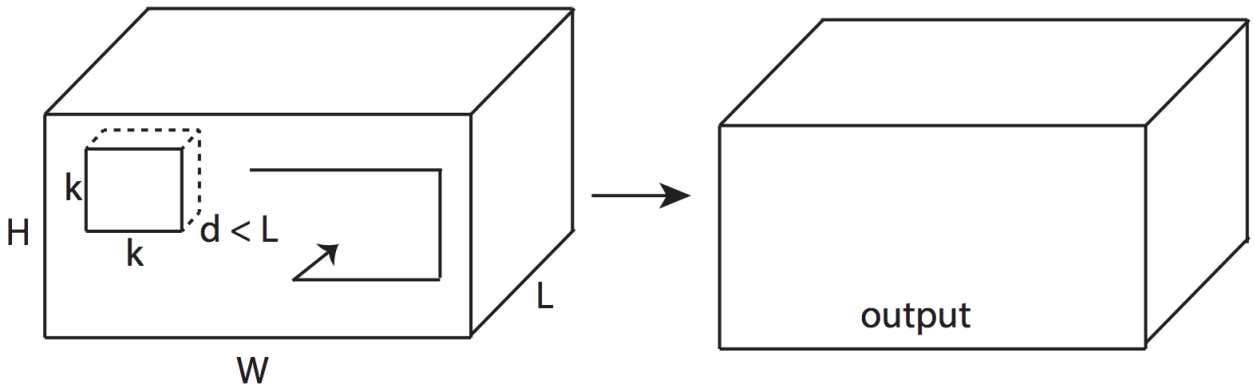



Рисунок 3.3 – 3D згортка

Прикладом вдалого застосування 3D згорток є архітектура **C3D** [14]. У цій архітектурі використовується 8 згорткових і 5 агрегувальних 3D шарів, 2 повнозв'язних шари (4096 нейронів у кожному) та softmax шар для передбачення класів. Кількість фільтрів у кожному з 8 шарів дорівнює відповідно 64, 128, 256, 256, 512, 512, 512, 512. Було показано, що згорткові фільтри 3x3x3 дають найкращий

результат. Агрегувальні шари мають розмір ядра 2x2x2 (окрім першого, який має розмір 1x2x2 аби уникнути раннього стиснення часового сигналу) і крок (stride) 1.

У цій роботі відео поділяється на сегменти з 16 кадрів. Для двох послідовних сегментів 8 кадрів є спільними (тобто перетинаються). Таких чином, вхідний об'єм згорткової мережі: $W \times H \times 3 \times 16$ ($W$ – ширина відео, $H$ – його висота, 3 канали кольору, 16 кадрів). Для побудови представлення (дескриптору відео) вихідні значення першого повнозв'язного шару усереднюються для всіх сегментів із 16 кадрів.

Найкращий результат, отриманий у [14], досягає точності 90.4% на UCF101 із використанням оптичного потоку, але без звукової доріжки. Теоретично CNN може моделювати необхідні залежності без оптичного потоку, але його наявність значно покращує результат на практиці. Це гірший результат, аніж при використанні багатопотокової моделі із CNN і RNN [32]. Також варто зазначити, що цей результат було досягнуто за допомогою трьох C3D, перша з яких була натренована на наборі даних Sports-1M [34], друга – на I380K (внутрішній (непублічний) великий набір даних, які використовували автори статті), третя – натренована на I380K, fine-tuned на Sports-1M. Ці моделі були використані для створення векторів представлення, на основі яких класифікацію було виконано за допомогою лінійного SVM.

Загалом отриманий результат непоганий, але варто відмітити, що натренована модель була не на UCF101, а на цих двох великих наборах даних. Параметри I380K не були опубліковані, а от Sports-1M містить 1.1 мільйони відео у 487 класах. Це означає, що в ньому у 100 разів більше відео, ніж у UCF101, і майже вп'ятеро більше класів. З цього можна зробити декілька висновків:

- Використовується CNN із значною кількістю параметрів. Менша нейромережа отримала б значно гірший результат, тому важливо мати декілька згорткових шарів із достатньою кількістю фільтрів, аби мережа мала змогу моделювати складні процеси, зображені на відео.
- Опублікований результат демонструє, що передавальне навчання добре працює на практиці. Натреновані на великих наборах даних (Sports-1M) моделі

використовувалися для створення представлень, і ці натреновані моделі застосовувалися на цільовому наборі даних (UCF101) без змін (тобто без додаткового навчання). Таким чином, можна бачити, що ці мережі навчилися виділяти ознаки так, що їх можна використати на іншому наборі даних і отримати досить добрий результат. Звісно, тут йдеться саме про класифікацію людських дій, а не про розпізнавання довільних відео взагалі (що також може працювати, але це потребує додаткового дослідження).

- Такий підхід показує також, що для навчання достатньо великої і глибокої CNN, яка необхідна для успішної класифікації, потрібна також велика кількість даних через більшу кількість параметрів, які додаються через наявність ще одного виміру в розмірності згорткового фільтру. Напряму на UCF101 модель C3D авторами не навчалась, лише її менший варіант. Натомість, архітектура з CNN і RNN досягала кращих результатів при тренуванні лише на UCF101. Таким чином, можна сказати, що *ефективність використання тренувальних даних (sample efficiency)* у архітектури CNN+RNN вища, у той час як для згорткових 3D мереж потрібні великі набори даних для досягнення потрібних результатів.

У [14] вводиться спеціальна архітектура CNN. Передавальне навчання реалізується за допомогою тренування на великих наборах даних із відео. У такого підходу є два недоліки: по-перше, ця архітектура має набагато менше шарів, ніж сучасні 2D CNN для класифікації зображень; по-друге, тренування проводиться з «чистого листа», у той час як в архітектурі з CNN+RNN згорткова нейромережа була попередньо натренована на ImageNet, тобто у C3D не використовуються «знання», отримані при тренуванні на великих наборах даних зображень.

У [35] пропонується вирішення цих проблем за допомогою архітектури **Two-Stream Inflated 3D ConvNet (I3D)**. Цей метод передбачає використання відомих архітектур 2D CNN із параметрами, натренованими на ImageNet. Автори пропонують розгортати (*inflate*; 2D фільтри (розміром $F \times F$) у тривимірні фільтри ($F \times F \times F$), просто додаючи часовий вимір. Для початкової ініціалізації параметри 2D фільтрів

повторюються $F$ разів упродовж часового виміру, і масштабуються діленням на $F$. Така ініціалізація еквівалентна дуплікації одного зображення багато разів для формування відео; тобто по суті виконується неявне тренування на ImageNet так, щоб агреговані активації на такому відео із одного повторюваного кадру були такими ж, як із одним цим вхідним кадром.

Таким чином, при такому підході можна використовувати дуже глибокі архітектури, які успішно працюють для розпізнавання зображень, і при цьому використовуються «знання», отримані 2D CNN при тренуванні на ImageNet. При тренуванні безпосередньо на наборі відео описана ініціалізація часового виміру дає гарні початкові значення, які пришвидшують навчання.

I3D використовує два потоки: одна CNN тренується на RGB зображення, а інша – на оптичних потоках. Зазначається припущення, що використання оптичного потоку підвищує точність через те, що CNN виконує одне пряме обчислення, у той час як методи знаходження оптичного потоку є рекурентними (виконується ітеративна оптимізація).

Архітектура I3D в наведених експериментах використовувала Inception-V1 [4] як базову модель, і більші часові сегменти (64 кадри / оптичні потоки, що еквівалентно 2.56 секунд відео з частотою 25 кадрів на секунду). Для двопотокової архітектури було отримано точність 93.4% на UCF101. Також наводить результат у 98.0% у разі попереднього навчання не лише на ImageNet, а і на великому наборі відео Kinetics [36]. Цей набір даних містить близько 300000 відеокліпів, що покривають 400 категорій. Таким чином, I3D – це архітектура, яка забезпечує гарний рівень точності при тренуванні на невеликих наборах відео, і цей рівень може бути покращено при попередньому тренуванні на більших обсягах даних.

Порівняно із сучасними 2D CNN, що використовуються для класифікації зображень, архітектура I3D все ще порівняно неглибока, бо містить лише 22 шари.

У [37] досліджуються більш глибокі 3D CNN на базі архітектури **ResNet** [5]. У цій роботі було експериментально досліджено архітектури з кількістю шарів від 18 до 200. Було показано, що ці архітектури надають достатній рівень точності на наборі даних Kinetics. Тобто зроблено висновок, що цей набір даних достатньо великий, аби

не відбувалося перенавчання навіть мереж із 200 шарами. Модифікована архітектура ResNet-101 (названа **ResNeXt-101**) при тренуванні лише на UCF101 мала точність 90.7%, що, знову ж таки, гірше, ніж у архітектурі з CNN і RNN, але при тренуванні ResNeXt-101 не використовувався оптичний потік. Також ця робота ще раз підтверджує ефективність передавального навчання: модель, попередньо натренована на Kinetics, дала точність 94.5% на UCF101. Однак варто зауважити одну деталь: у статті зазначається, що із UCF101 було видалено кадри, на який нема дій. Це значна модифікація, оскільки вона *маскує основний недолік 3D CNN*. Архітектури зі згортковими 3D нейромережами обробляють окремі сегменти відео, і результати передбачень на окремих сегментах усереднюються; таким чином, якщо значна частина відео не містить дії, що повинна детектуватися, результат буде неправильним. На мою думку, модель машинного навчання для загального розпізнавання відео повинна бути стійка до таких даних без штучної передобробки.

Загалом, основним результатом [37] є висновок про те, що набір даних Kinetics має достатній розмір і різноманіття, щоб тренувати згорткові 3D мережі так, щоб вони могли виступати екстракторами представлень загального призначення та могли бути основою для передавального навчання; тобто Kinetics в цьому сенсі є аналогом ImageNet, який надає таку значну базу для 2D CNN.

Насамкінець, варто відмітити, що сучасні архітектури 3D CNN показують чудові результати. Однак, для їх досягнення необхідне навчання моделей на великих наборах даних через їх меншу ефективність використання тренувальних даних, що накладає додаткові вимоги на апаратне забезпечення (з точки зору часу та вартості). Також велика кількість параметрів 3D CNN вимагає значної кількості пам’яті на GPU, що є проблемою для сучасних відеокарт, найкращі з яких зазвичай мають 11-16 Гб відеопам’яті. Таким чином, відтворення результатів тренування глибоких 3D CNN на великих наборах даних, таких як Kinetics, вимагає великої кількості часу на навчання моделей, і кластеру дуже значного розміру.

## 3.3 Обчислення оптичного потоку за допомогою згорткових нейромереж

Задача обчислення оптичного потоку довгий час вирішувалася за допомогою класичних методів комп'ютерного зору, таких як згаданий раніше алгоритм Лукаса-Канаде. Але останніми роками з'явилися роботи, що досліджують використання нейромереж для вирішення цієї задачі.

**Постановку задачі навчання нейронної мережі для обчислення оптичного потоку** можна сформулювати так. Маючи два кадри відеопослідовності, потрібно апроксимувати рух кожного пікселя з першого кадру. Виходячи із припущення, що інтенсивності пікселів константі між кадрами, можемо записати:

$$I(x, y, t-1) = I(x+u, y+v, t), \tag{3.2}$$

де $I$ – інтенсивність зображення як функція від простору $(x, y)$ і часу $t$;

$(u, v)$ – координати вектору переміщення, або оптичний потік, який необхідно апроксимувати.

Це означає, що якщо взяти перше зображення $I(x, y, t-1)$ і перемістити пікселі на $(u, v)$, то отримаємо наступне зображення $I(x+u, y+v, t)$. Функцію, яку має реалізувати нейронна мережа, можна записати так:

$$(u, v) = f(I_{t-1}, I_t) \tag{3.3}$$

Сучасні підходи до вирішення цієї задачі за допомогою нейронних мереж описано у [15, 38, 39]. Значний час проблемою була недоступність тренувальних даних. Для їх створення потрібно визначити точне переміщення кожного пікселя на зображенні, і ця задача не може бути вирішена людьми. Ця проблема була вирішена за допомогою комп'ютерної графіки, інструментами якої було згенеровано відеопослідовності. Оскільки при такому процесі переміщення описуються певними трансформаціями (поворотами, перенесеннями тощо), то точно відомо, як змістився певний піксель у наступний момент часу.

При тренуванні архітектури FlowNet [15] використовувався більш простий синтетичний набір даних, але було показано, що результати добре узагальнюються на

більш реалістичні відео. Наявний також більш складний набір – MPI Sintel Flow Dataset [40], автори якого виділили оптичні потоки із анімованого 3D короткометражного фільму із відкритим вихідним кодом.

Загалом, архітектура FlowNet показала кращі результати, аніж найкращі класичні методи. Результуючі зображення оптичного потоку є більш точними, із чіткішими контурами. Однак варто відмітити, що це досить велика модель, яка має близько 32 мільйонів параметрів; у [38] було досліджено іншу архітектуру, яка має всього 1.2 мільйони параметрів, працює швидше і досягає прийнятного рівня точності.

На даний момент нейромережою, яка досягає найвищої точності, є FlowNet 2.0 [39]. Це покращення архітектури FlowNet [15], яке досягло зменшення помилки апроксимації більш ніж на 50% із лише незначним сповільненням.

## 3.4 Висновки

Таким чином, можна виділити дві основні архітектури, що застосовуються для класифікації відео: такі, що використовують двовимірні згорткові та рекурентні нейронні мережі, та ті, які застосовують згорткові 3D мережі.

У багатопотокових архітектурах зазвичай виділяють такі потоки: просторовий (послідовність кадрів), часовий (рух, виражений оптичним потоком), і звуковий. Кожен потік аналізується окремо, і потім виконується їх синтез певним способом задля отримання результату класифікації.

У архітектурах із CNN і RNN згорткові мережі використовуються для отримання вектора представлення кадрів і оптичних потоків, і ці вектори подаються на вхід рекурентної нейромережі. Таким чином, RNN отримують на вхід послідовність векторів представлень.

За допомогою архітектур з тривимірними згортковими нейронними мережами отримують гарні результати, але такі мережі мають достатньо велику кількість параметрів, що вимагає значного часу навчання, великого обсягу пам’яті на GPU, а відповідно їх навчання видається менш економічно ефективним. Також тренування

таких нейромереж вимагає великих наборів даних, і це обмежує їх використання на невеликих наборах даних, хоча це вирішується за допомогою застосування передавального навчання.

Сучасні методи обчислення оптичного потоку за допомогою нейронних мереж переважають класичні алгоритми комп’ютерного зору. Їх використання дозволяє отримати більш точні апроксимації оптичного потоку за коротший час.

# 4 ЗАПРОПОНОВАНИЙ МЕТОД КЛАСИФІКАЦІЇ ВІДЕО

У двох попередніх розділах було описано теоретичні основи вирішення задачі класифікації відео засобами машинного навчання, та проаналізовано опубліковані методи для вирішення цієї задачі. Було показано, що основними підходами є архітектури із 2D CNN і RNN, а також методи з 3D CNN без використання RNN. Запропонований метод ґрунтуватиметься на першому підході. Обґрунтуємо цей вибір.

Тривимірні згорткові архітектури є новішими, і вже показують гарні результати. Однак важливою технічною проблемою є складність тренування таких моделей через значні вимоги до обчислювальних ресурсів. Іншою проблемою є необхідність використання дуже великих наборів даних, аби мати можливість навчити 3D CNN, що мають велику кількість параметрів через наявність третього (часового) виміру. Ці дві проблеми демонструють *нижчу ефективність* таких моделей, бо на невеликих наборах даних можна швидше отримати кращі результати за допомогою комбінації 2D CNN і RNN. У цій роботі експериментальне випробування запропонованого методу провадитиметься на відносно невеликому масиві відео з огляду на доступне апаратне забезпечення. До того ж, цінними є методи, які дозволяють отримати гарні результати із невеликою кількістю даних. Розмітка даних людьми є коштовною, і якщо певний метод застосовується для вирішення певної спеціалізованої задачі, наприклад, в індустрії, а не лише для роботи із публічним набором даних, то часто кількість наявних промаркованих відео не досягатиме мільйонів.

Ефективність тривимірних згорткових архітектур є більш технічним або економічним питанням. Збільшити кількість тренувальних даних можна отримати технічними способами, або витративши більше коштів на їх створення людьми. Також майбутні вдосконалення таких методів, безперечно, покращать їх ефективність як з точки зору обчислювальної продуктивності (менше параметрів, краща

оптимізація коду) і ефективності використання даних так само, як це спостерігалось в процесі еволюції згорткових нейромереж для розпізнавання зображень.

Однак, основною проблемою, на мою думку, є *концептуальна*. Ця проблема пов'язана із тим, як улаштовані згорткові нейромережі.

Згорткові шари детектують важливі ознаки на зображеннях. Глибші шари (ті, які ближче до входу), детектують простіші характеристики (як краї, градієнти кольору тощо), у той час як вищі об'єднують ці прості ознаки в більш складні. Повнозв'язні шари в кінці нейромережі комбінують ці дуже високорівневі ознаки та повертають передбачення. Але варто зазначити, що верхні шари комбінують ознаки з нижчих як зважену суму: активації попереднього шару множаться на ваги нейронів поточного шару та додаються перед передачею до нелінійної функції активації. Ідеологічно, це не найкраща ідея. Якщо розглянути зображення обличчя як приклад, то це не просто певний невпорядкований набір ознак, таких як очі, ніс тощо. Аби ці ознаки представляли обличчя, вони повинні мати певне просторове розміщення та орієнтацію, тобто існує *залежність переносу та повороту* між простими ознаками (очі, ніс), які формують більш складну (наявність обличчя на зображенні). Присутність пари очей, рота і носа на зображення не означає наявність обличчя; потрібно знати, як ці об'єкти орієнтовані один щодо іншого. *Згорткові мережі не моделюють такі залежності.* Підходом до вирішення цієї проблеми, який застосовується в CNN, є агрегаційні шари, які зменшують просторову розмірність даних при їх обробці послідовними шарами мережі, таким чином збільшуючи «поле зору» нейронів вищих шарів, дозволяючи їм детектувати ознаки у більших регіонах вхідного зображення. Підхід із використанням агрегаційних шарів призводить до втрати цінної інформації. *Внутрішнє представлення даних згорткової нейромережі не враховує важливих просторових ієрархій між простими та складними об'єктами.* Спробою вирішити цю проблему є капсульні нейромережі [6, 7].

Таким чином, хоча згорткові нейромережі дуже добре працюють на практиці, у них є фундаментальні недоліки; успіх їх використання залежить від даних. Ця проблема також проявляється при використанні 3D згорткових нейронних мереж для аналізу відео. Коли 3D фільтри «сканують» часовий вимір, то виникає та ж сама

проблема, коли детектовані на окремих сегментах часового виміру ознаки об’єднуються в більш складні без урахування їх взаємного розташування.

Ще однією проблемою 3D CNN при застосуванні їх до аналізу відео є те, що вони обробляють не повне відео на відміну від 2D CNN для класифікації зображень, які отримають на вхід повне зображення. На практиці відео розбивається на сегменти, і передбачення виконуються для кожного сегмента окремо, і потім результати об’єднуються (наприклад, усереднюються). Це необхідно через те, що в іншому випадку розмірність вхідних даних була б занадто великою. Унаслідок, якщо аналізується відео, на якому дія або процес, що повинні бути класифіковані, відбувається лише на невеликому сегменті відео, загальний результат класифікації буде неправильним.

На мою думку, ці концептуальні проблеми тривимірних згорткових нейромереж є досить значними. Вони добре працюють на практиці, але публічні набори даних із відео зазвичай мають короткі ролики, які зображають безпосередньо певну дію. Такі дані маскують недоліки згорткових нейромереж, які в загальному випадку є дуже обмеженими.

*З урахуванням зазначеного аналізу, запропонована архітектура використовуватиме рекурентні нейронні мережі для моделювання послідовностей.*

## 4.1 Загальна архітектура

Як було проаналізовано, запропонована у [32] архітектура є майже оптимальною: моделюються всі три потоки (просторовий, часовий і звуковий), використовуються рекурентні нейромережі для моделювання послідовностей, застосовується передавальне навчання (2D CNN натренована на ImageNet). Але було відмічено один недолік: оскільки у цьому підході кожен потік класифікується окремо, по суті модель, що виконує синтез, незалежна від характеристик, отриманих у кожному потоці, а просто поєднує передбачення найкращим чином.

Ключовою ідеєю вдосконалення є поглиблення використання представлень (representation learning). Замість того, щоб класифікувати кожен потік окремо, будемо

брати представлення, отримане кожним потоком. Ці представлення будуть об’єднуватися у результат класифікації окремою моделлю синтезу (fusion model). Дещо схожий підхід було наведено у [41], але там було отримано порівняно низьку точність класифікації. Запропоновану архітектуру зображено на рис. 4.1.

Таким чином, для просторового (spatial) потоку, представленого послідовністю кадрів маємо: пікселі (RGB) кожного кадру подаються на вхід згортковій мережі, яка повертає *вектор представлення* (з передостаннього шару перед тим, який повертає результат класифікації) *кадру*. Вектори представлень усіх кадрів відео подаються на вхід рекурентній нейронній мережі, яка повертає *вектор представлення відео* (який береться з виходу останньої комірки RNN).

Для часового потоку на вході маємо знову ж таки кадри у вигляді пікселів. Для пар кадрів обчислюється оптичний потік з одиничним кроком (тобто між 1 і 2 кадрами, потім між 2 і 3 тощо). Ці оптичні потоки конкатенуються в групи по N потоків. Далі окрема CNN повертає вектор представлення для групи оптичних потоків. Послідовність цих векторів представлення подається на вхід RNN, яка, в свою чергу, повертає вектор представлення відео.

Звуковий потік поділяється на сегменти певної довжини (наприклад, 1 секунда). Для кожного сегменту обчислюється спектрограма, за якою інша згорткова нейромережа обчислює вектор представлення. Знову ж таки, послідовність векторів представлень подається на вхід рекурентної нейронної мережі цього потоку, відповідно до якої визначається вектор представлення відео.

Як бачимо, підхід обробки кожного потоку можна узагальнити так: вхідні дані (кадри або звукова доріжка) → передобробка (виділення оптичного потоку, групування, обчислення спектрограми) → CNN → вектори представлень кадрів → RNN → вектор представлення відео.

Таким чином, у кінці ланцюжка обробки кожного потоку маємо вектори представлень відео. Ці три вектори з кожного потоку характеризують різні ознаки відео (просторові, рух, аудіо). *Разом вони є компактним представленням різних ознак відео.*

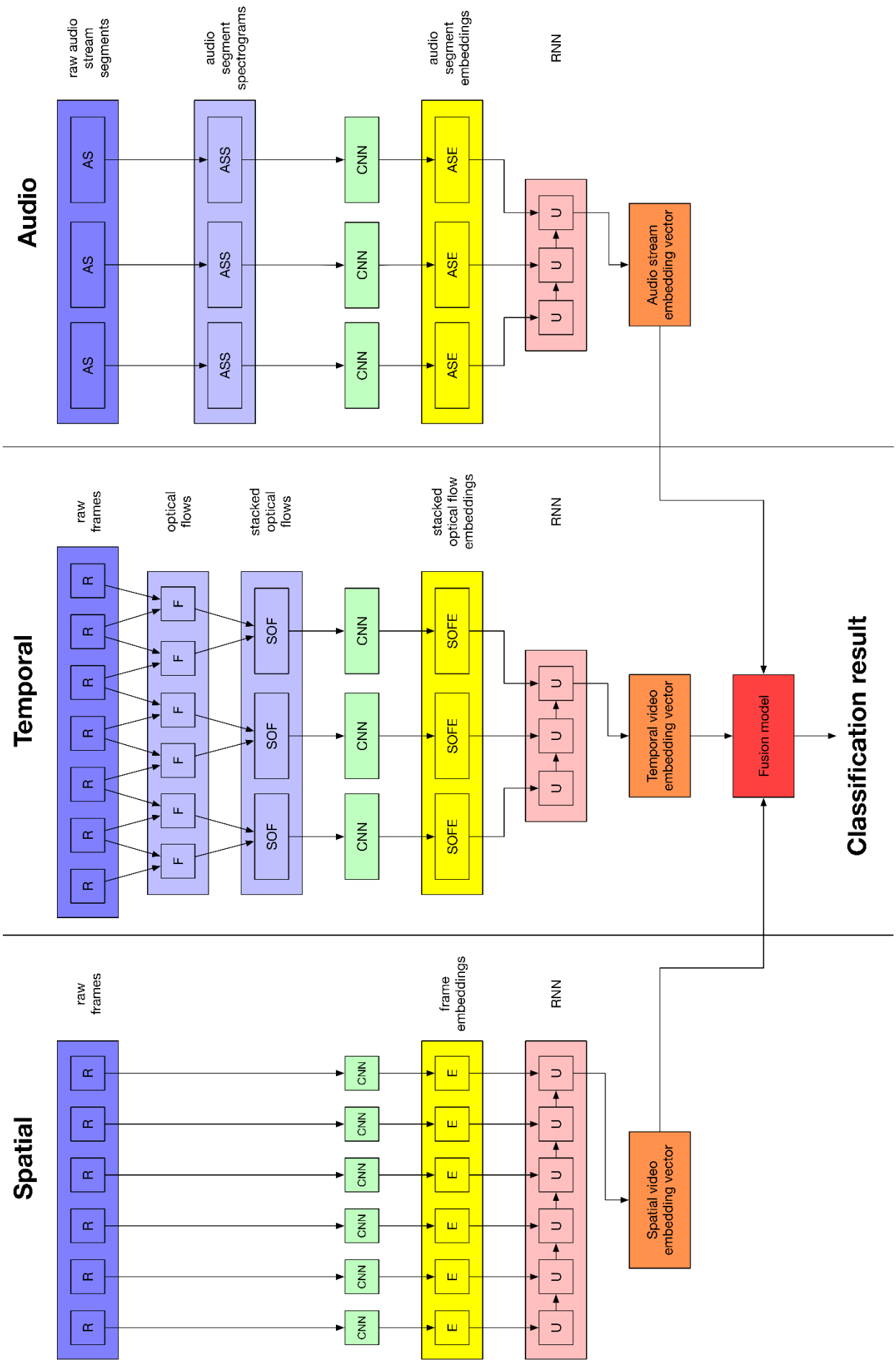



Рисунок 4.1 – Загальна архітектура запропонованої моделі класифікації відео

Таким чином, замість результату класифікації кожного потоку окремо, маємо представлення відео загалом. Далі це компактне представлення можна подати на вхід певній моделі синтезу, яка поверне ймовірності належності відео до кожного з класів.

Це компактне представлення можна гнучко обробляти: використовувати різні моделі синтезу або застосовувати як вхідні дані якоїсь іншої моделі. Наприклад, якби стояла задача класифікації відео та якогось його текстового опису, певна модель синтезу могла би приймати на вхід конкатенацію векторів представлення тексту та відео. Також це представлення можна використовувати для виявлення аномалій.

## 4.2 Архітектури нейромереж окремих компонентів

Тепер опишемо конкретні моделі, які використовуватимуться у цій архітектурі.

Як *згорткова нейронна мережа для екстракції векторів представлення кадрів* застосовується **InceptionResNetV2** [42]. Це одна з найкращих на даний час архітектур, яка на ImageNet досягає точності 80.4% top-1, 95.3% top-5. До того ж, вона досить ефективна: у старішій VGG19 [3], яка досягала точності 72.7%, близько 143.5 мільйонів параметрів; у InceptionResNetV2 – приблизно 56 мільйонів.

*CNN для груп оптичних потоків* не така, як для кадрів, оскільки використання готової нейромережі для зображень неможливе через різні кількості каналів. За основу було взято запропоновану у [29] архітектуру з незначними модифікаціями. Результат виглядає так:

conv1 (7x7x96; stride 2; pooling 3x3; norm) → conv2 (5x5x256; stride 2; pooling 3x3) → conv3 (3x3x512; stride 1) → conv4 (3x3x512; stride 1) → conv5 (3x3x512; stride 1; pooling 3x3) → full6 (4096; dropout) → full7 (2048; dropout) → softmax

У цих позначеннях FxFxN для згорткових (conv) шарів означає N фільтрів розміру FxF. Крок (stride) відноситься до згорткових шарів. Максимізаційна агрегація (max pooling) всюди виконується шарами розміру 3x3 із кроком 2 (важливим є перетин (overlap), який є при такому виборі кроку; такі агрегувальні шари використовуються у [2-4]). Нормалізація (norm) у [29] використовує local response normalization. У запропонованому підході використовується пакетна нормалізація

(batch normalization) [43]. Пакетна нормалізація дозволяє прискорити навчання моделі та має деякий регуляризаційний ефект через додавання певного шуму. Для повнозв'язних шарів (full) вказано кількість нейронів. Dropout [44] позначає проріджування нейромережі для зменшення перенавчання (тобто це спосіб регуляризації). Таким чином, із переодостаннього шару можна отримати 2048-вимірне представлення.

Для *обчислення оптичного потоку* використовується архітектура FlowNet 2.0 [39], яка на даний час має найкращу точність.

Для визначення векторів представлень аудіо застосовується модель VGGish [12].

Є значний вибір різних архітектур рекурентної мережі для вирішення даної задачі. Оскільки послідовності досить довгі, то звичайна RNN не підійде, і варто використовувати LSTM або GRU. Експериментальним шляхом (результати наведено у розділі 7) було визначено архітектуру, яка дає найкращі результати на UCF101: це два двонапрямлених шари GRU (stacked bidirectional GRU) із розмірністю прихованого стану кожної комірки, рівною 512. Також експериментальним шляхом було визначено, що використання SVM замість softmax на класифікаційному шарі дає найкращу точність класифікації.

Як модель синтезу, яка приймає на вхід вектори представлень із усіх трьох потоків і повертає ймовірності належності відео кожному з класів, використовується лінійна SVM, яка у експериментах дала найкращий результат. Також тут можна використати логістичну регресію або нейронну мережу.

## 4.3 Навчання моделі

Для тренування запропонованої багатопотокової архітектури є два підходи. Перший – це *тренувати всі разом*, цей підхід має назву end-to-end training, або joint training. Інтуїтивно, цей підхід має давати кращі результати. Але тренування такої моделі вимагатиме великої кількості пам'яті GPU. До того ж, велика кількість параметрів, що оптимізуються, не лише сповільнює тренування, але і робить його

набагато менш стабільним. Ключовим спрощенням є *тренування кожного потоку окремо*. У [45] було зазначено, що тренування усієї моделі разом дає лише незначний приріст точності. До того ж, тренування кожного потоку окремо збільшує гнучкість архітектури, дозволяючи замінити окремі компоненти без повторного навчання всієї моделі.

Таким чином, при тренуванні кожного потоку окремо оптимізується значення функції втрат для цільової задачі класифікації. Використовується припущення, що в цьому випадку знаходяться оптимальні вектори представлень, які найкращим чином описують дані.

Fine-tuning моделі для звукового потоку (VGGish) не проводився, оскільки використовуваний набір відео має достатньо мало прикладів для навчання. Застосувалися ваги, отримані при тренуванні на AudioSet [46], який є дуже великим і якого є більш ніж достатньо, аби модель могла обчислювати вектори представлень для майже будь-яких звукових даних.

Модель для кадрів (InceptionResNetV2) також не дотреновувалася на цільовому наборі даних, використовувалися ваги натреновані на ImageNet. Було експериментально перевірено, що fine-tuning на UCF101 зменшує якість векторів представлень, що призводило до зниження точності класифікації.

Згорткова мережа для оптичних потоків має тренуватися з нуля, бо це спеціалізована архітектури, для якої немає натренованих параметрів у відкритих джерелах.

З урахуванням написаного, процес навчання включає такі кроки (можуть виконуватись у будь-якому порядку):

- Тренування RNN просторового потоку.
- Тренування CNN часового потоку, потім тренування RNN цього потоку.
- Тренування RNN для аудіо.

Після виконання цих кроків обчислюються вектори представлень для всіх відео у наборі, і здійснюється тренування моделі синтезу (лінійна SVM), яка приймає на

вхід сконкатеновані вектори представлень усіх потоків, а повертає ймовірності належності відео до наявних класів.

## 4.4 Деякі аспекти вдосконалення методу

### 4.4.1 Доповнення даних

Потужною технікою вдосконалення навчання моделей машинного навчання є **доповнення даних (augmentation)**. Архітектури глибинного навчання зазвичай мають завелику кількість параметрів, що, в свою чергу, потребує великої кількості даних для їх навчання. Ручна розмітка даних є коштовною, але можна легко збільшити кількість наявних даних штучно за допомогою технік доповнення даних.

Ці техніки полягають у створенні нових навчальних прикладів на основі існуючих, що виконується за допомогою набору трансформацій. Наприклад, для зображень можна здійснювати такі перетворення: повороти, розмиття (blur), обрізання (crop), зашумлення, переноси, перевороти відносно осі, наближення певних областей, масштабування, розтягування тощо. Виконання таких трансформацій із довільними параметрами дає можливість створити по суті нескінченну кількість даних. *Це дозволяє тренувати значно складніші класифікатори із меншою кількістю даних.* Зрозуміло, що успіх цього процесу все одно залежить від вхідних даних – якщо вони недостатньо різноманітні, або погано репрезентують дані, які має моделювати модель, доповнення даних не допоможе вирішити всі проблеми.

Ще одна важлива роль доповнення даних – це *регуляризація*. Стохастичні перетворення навчальних прикладів запобігають перенавчанню, бо за їх наявності модель не має можливості «запам’ятати» тренувальний набір даних. У результаті моделі краще узагальнюються та є більш стійкими до природніх перетворень даних, які мають розпізнаватися, таких як шум на фотографіях.

Описаний підхід стосувався трансформування навчальних даних. Але гарною ідеєю є також **доповнення даних під час класифікації (test-time augmentation – TTA)**. Тобто під час класифікації для кожного прикладу створюється набір прикладів шляхом перетворення вихідного прикладу, виконується класифікація кожного

прикладу з цього набору, а результуюче передбачення отримується шляхом усереднення. На практиці це дозволяє збільшити точність класифікації.

Зазначені техніки застосовувалися в запропонованому методі, тому що використовуваний набір відео (UCF101) є порівняно невеликим, тому доповнення даних має сенс.

### 4.4.2 Змагальне тренування

Як вже було зазначено, для сучасних моделей розпізнавання графічних даних існує проблема змагальних атак (adversarial attack) [8], коли додавання шуму спеціальним чином може призвести до абсолютно неправильної класифікації прикладів моделлю. До того ж, змагальне зображення може виглядати майже ідентично оригіналу (з точки зору людини). Було показано, що можливо підібрати змагальні приклади модифікуючи лише один піксель зображення [47]. Це показує вразливість сучасних підходів розпізнавання.

Створення змагальних прикладів є задачею оптимізації, коли підбирається шум таким чином, аби *максимізувати помилку моделі*. Або, сформулювавши інакше, можна підбирати шум аби *мінімізувати впевненість моделі* (тобто ймовірності в передбаченні).

Таким чином, змагальне тренування полягає в навчанні моделі на таких змагальних прикладах. Це робить її більш стійкою (robust) до таких атак.

### 4.4.3 Використання капсульних мереж

Як зазначалося, згорткові нейронні мережі мають значні обмеження через втрату інформації агрегувальними шарами, що призводить до того, що CNN не враховують просторові ієрархії між простими та складними об’єктами. Спробою вирішити цю проблему є капсульні нейромережі (CapsNet) [6, 7], які явно моделюють відношення між об’єктами (переноси та повороти), відповідно зберігаючи ієрархічні відношення позиції між об’єктами та їх частинами.

*На мою думку модель із властивостями, подібними до CapsNet, могла би формувати такі вектори представлення, що моделювання їх послідовності за*

*допомогою рекурентних нейронних мереж було би подібно до апроксимації руху оптичним потоком.*

На жаль, у даний час наявні підходи до тренування капсульних нейромереж є дуже повільними, і такі мережи показують гарні результати лише на невеликих наборах даних.

### 4.4.4 Адаптивна вибірка

Зазвичай часова розмірність вхідних даних зменшується, оскільки рекурентні нейронні мережі складніше тренувати з довшими послідовностями. Наприклад, замість стандартної частоти кадрів на відео у 25 кадрів на секунду, вибирається 5. Стандартним підходом є вибірка через однакові інтервали, тобто в цьому випадку братиметься кожний п’ятий кадр (перший, п’ятий тощо).

Але оскільки маємо оптичний потік, пропонується адаптивний метод семплювання, який враховував би рух. Оптичний потік є набором векторів переміщення; довжина цього вектора відповідно є відстанню, на яку перемістився піксель між кадрами. Таким чином, довжина вектору переміщення також є і швидкістю за одиницю часу (один кадр). *Відповідно ці довжини характеризують «активність» руху між кадрами.* Якщо в певному сегменті відео об’єкти зміщаються мало, то довжини векторів будуть малими. Це означає, що такий сегмент містить менше інформації ніж той, на якому були зафіксовані вищі швидкості переміщення пікселів (наприклад, швидко проїжджаюча машина). Таким чином, *замість константної частоти вибірки має сенс адаптувати частоту відповідно до швидкості руху пікселів.* Наприклад, якщо на 30-секундному кліпі нічого не відбувається протягом 25 секунд, а потім машина проїжджає в певному 5-секундному інтервалі, то можна брати частоту вибірки, що дорівнює 1 кадру на секунду в тих сегментах, де руху нема, а в тих сегментах, коли об’єкт рухається, брати більшу (аж до максимальної у 25 кадрів на секунду) частоту вибірки. Як міру «активності» руху між кадрами можна взяти середню довжину векторів переміщення пікселів. Таким чином, *модель детальніше «розглядатиме» сегменти з більш «активним» рухом*.

Ефектом також є те, що у сформованій таким чином послідовності кадри з рухом будуть представлені більшою кількістю прикладів.

Недоліком цього методу є те, що він змінює сприйняття швидкості руху на відео. Сегменти із більшою кількістю вимірів виглядатимуть (порівняно з іншими) так, ніби вони були відзняті уповільненою зйомкою.

На практиці довжина часового виміру даних на вході рекурентної нейромережі задається фіксованою. Відповідно, за допомогою адаптивної вибірки можна вибрати цю фіксовану кількість прикладів так, аби сегменти відео, що мають більшу кількість інформації, були представлені й більшою кількістю кадрів.

Опишемо більш формально. Нехай є цільова кількість кадрів $N$, а відео довжиною $t$ секунд містить $L$ кадрів із постійною частотою кадрів $f = L/t$ (кадрів на секунду). Розіб'ємо відео на $K$ сегментів довжини $T$, наприклад, при $T = 1$ секунда кожен сегмент міститиме $S = f$ кадрів, де $S$ – кількість кадрів у сегменті. Обчислимо оптичні потоки між усіма послідовними парами кадрів у кожному сегменті (тобто між першим і другим, другим і третім тощо). Для кожної пари отримаємо набір векторів переміщення кожного пікселя $\{d(x, y)\}, x = [1, h], y = [1, w]$, де $h$ – висота відео, $w$ – ширина відео. Обчислимо середнє значення довжин усіх векторів:

$$\bar{v} = \frac{1}{w \times h} \sum_{x,y} |d(x, y)| \tag{4.1}$$

Назвемо $\bar{v}$ швидкістю (середньою) руху пікселів між двома кадрами. Далі обчислимо середнє значення швидкостей руху між усіма парами послідовних кадрів у сегменті:

$$\bar{v}'(k) = \frac{1}{S} \sum_s \bar{v}(s), \tag{4.2}$$

де $\bar{v}(s)$ – швидкість руху пікселів між $s$-тим і наступним кадром.

Таким чином, $\bar{v}'(k)$ – це середня швидкість руху пікселів у $k$-ому сегменті з $S$ кадрів. У нашому припущенні ця величина пропорційна кількості інформації у сегменті. Відповідно вибиратимемо кількість кадрів у сегменті пропорційно цій величині, щоб сегменти із більшою «активністю» руху мали більше кадрів у вибірці.

Корисно накласти обмеження на мінімальну кількість кадрів, що вибирається із сегмента. Назвемо цю величину $m$. Тоді з кожного сегмента безумовно виберемо

$m$ кадрів, а кількість додаткових визначатиметься величиною $\bar{v}'$. Тобто адаптивна вибірка відбере сумарно $\widetilde{N} = (N - m \times K)$ кадрів. Далі необхідно визначити, скільки саме буде відібрано кадрів із кожного сегменту. Позначимо цю величину як $N'(k)$, де $k$ – номер сегмента (усього сегментів $K$). Таким чином, потрібно забезпечити, щоб $N'(k) \sim \bar{v}'(k)$ (тобто кількість вибраних кадрів із сегмента була пропорційна швидкості руху пікселів у сегменті), а також:

$$\sum_{i=1}^{S} N'(k) = \widetilde{N} \tag{4.3}$$

Один із способів зробити це є таким. Є множина швидкостей руху у сегментах $\{\bar{v}'(k)\}$ ($k \in [1; K]$). Можна нормувати ці значення, щоб вони були на проміжку від 0 до 1, а їх сума дорівнювала одиниці. Наприклад, це можна зробити так (оскільки значення $\bar{v}'(k)$ невід'ємні:

$$\bar{v}''(k) = \frac{\bar{v}'(k)}{\sum_{k=1}^{K} \bar{v}'(k)} \tag{4.4}$$

Альтернативним варіантом є застосування softmax.

Таким чином, маємо множину «ваг»: $\{\bar{v}''(k)\}$. Відповідно, із кожного сегмента вибираємо кількість кадрів, пропорційну обчисленій «вазі» сегменту: $N'(k) = \widetilde{N} \times \bar{v}''(k)$ (потрібно враховувати реалізацію округлення, щоб значення $N'(k)$ в сумі давали $\widetilde{N}$). Підсумковою кількістю кадрів, що вибирається із $k$-го сегменту, є $(N'(k) + m)$. Конкретні кадри можна вибрати із регулярним кроком, скажімо, рівним $\left\lfloor \frac{f}{N'(k)} \right\rfloor$.

Розглянемо простий приклад. Скажімо, маємо кліп довжиною $t = 3$ секунди із частотою $f = 25$ кадрів на секунду, отже $L = 75$. Цільова кількість кадрів для вибірки $N = 15$ (тобто зменшення розмірності у 5 разів). Довжину сегменту виберемо $T = 1$ секунда, отже $K = 3$. Мінімальну кількість кадрів на сегмент покладемо $m = 1$. Після обчислення оптичних потоків, отримали такі середні швидкості руху в сегментах: $\{\bar{v}'(k)\} = \{0; 4; 12\}$. Після вибірки мінімальної кількості кадрів із кожного сегменту, залишається $\widetilde{N} = N - m \times S = 15 - 1 \times 3 = 12$ кадрів для вибірки за допомогою адаптивного методу, тобто $\frac{\widetilde{N}}{K} = \frac{12}{3} = 4$ на сегмент. Після нормування отримали: $\{\bar{v}''(k)\} = \{0.0; 0.25; 0.75\}$. Вибираючи $\widetilde{N} \times \bar{v}''(k)$ із кожного сегмента, отримаємо:

$\{N'(k)\} = \{0; 3; 9\}$ (кількості кадрів, вибраних адаптивним методом із кожного із 3 сегментів). Враховуючи мінімальні кількості вибраних кадрів, підсумкові кількості є такими: $\{1; 4; 10\}$. Таким чином, у першому сегменті, наприклад, можна вибрати перший кадр. У другому крок дорівнює $\left\lfloor \frac{f}{N\prime(k)} \right\rfloor . = \left\lfloor \frac{25}{4} \right\rfloor = 6$, тому виберемо кадри із номерами 1, 7, 13, 19.

### 4.4.5 Механізм уваги

Механізм уваги у нейронних мережах частково базується на механізмі візуальної уваги у людей. Цей біологічний механізм добре вивчений, і хоча існують декілька різних моделей, вони всі зводяться до того, що увага – можливість фокусуватися на певному регіоні зображення та розглядати його з «великою розподільчою здатністю», при цьому сприймаючи навколишнє зображення із «низькою розподільчою здатністю», і зміщувати точку фокусу з часом.

Цей механізм уваги можна реалізувати за допомогою повнозв'язного шару такої ж довжини, як і у вхідного вектора. Таким чином, вихідний вектор матиме таку ж довжину, і кожне значення у ньому буде зваженою комбінацією всіх значень вхідного вектора. Ці вихідні значення шару уваги подаватимуться на вхід рекурентній нейромережі, яка повертатиме шуканий результат. При тренуванні оптимізуватимуться параметри як основної моделі, так і шару уваги, відповідно модель навчатиметься, на чому фокусуватися для покращення класифікації (або вирішенні іншої задачі).

Основною перевагою механізму уваги є можливість інтерпретувати результати, що повертає модель. Наприклад, у моделі, що виконує переклад з однієї мови на іншу, візуалізувавши ваги шару уваги, можна побачити, що модель враховує артикль і слово одночасно.

У деяких задачах NLP, наприклад, машинному перекладі, заміна рекурентних моделей на механізм уваги навіть дає кращі результати [48].

Але на відміну від людської уваги, яка призначена для економії обчислювальних ресурсів, механізм уваги в нейромережах збільшує обчислювальну

складність, оскільки повнозв'язний шар із розмірністю вхідних і вихідних векторів $N$, має $N^2$ параметрів.

## 4.5 Виявлення аномалій

Було зазначено, що запропонований метод, заснований на вивченні представлень, дає велику гнучкість. Наприклад, на основі нього можна будувати складніші моделі.

Однією з важливих задач аналізу відео є виявлення аномалій. Ця задача виникає, наприклад, у системах спостереження, коли потрібно виявити небезпечну ситуацію або вторгнення на територію. Цю задачу простіше розглядати як задачу навчання без учителя (unsupervised learning), оскільки відсутні тренувальні дані. У цій ситуації бажано мати модель, яка може відрізнити нестандартну ситуацію, що зображена на відео. Наприклад, на записах із камер спостереження на дорозі більшість часу рух відбувається нормально, але часом трапляються ДТП. Модель, що виконує виявлення аномалій, повинна вміти відрізняти такі ситуації.

Сучасний підхід до виявлення аномалій часто базується на популярних нині **автокодувальниках (autoencoder)**, схема якого зображена на рис. 4.2.

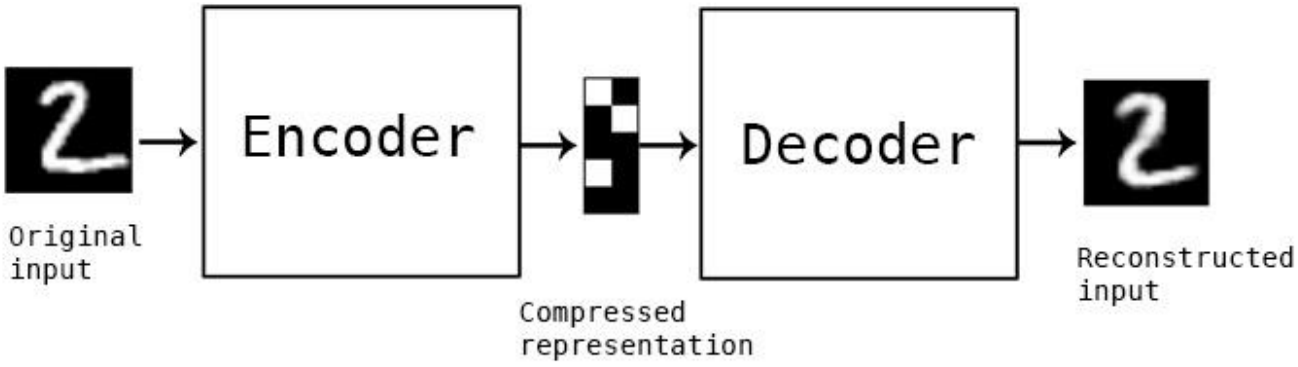



Рисунок 4.2 – Схема автокодувальника

На вході є тензор високої розмірності, який, проходячи через кодувальник (encoder), перетворюється на стиснене представлення. Потім декодувальник (decoder) відтворює вхідний тензор із цього представлення. Тренування відбувається так, що

параметри кодувальника та декодувальника оптимізуються таким чином, аби після проходження автокодувальника вхідні дані були відтворені максимально точно. Відповідно функція втрат, яка мінімізується, вимірює відмінність між вхідним і вихідним тензорами.

Автокодувальник, як і будь-яка нейронна мережа, апроксимує певну функцію. Існує дещо інший різновид – **варіаційний автокодувальник (variational autoencoder – VAE)** [49]. VAE вивчають параметри статистичного розподілу, який моделює дані. Якщо зробити вибірку із цього розподілу, можна отримати нові приклади (але схожі на тренувальні дані); таким чином, VAE – генеративна модель.

Загалом, можна стверджувати, що автокодувальник – це метод стиснення даних, вивчення представлень, або зменшення розмірності.

При стисненні даних автокодувальником, очевидно, відбувається втрата інформації. Але вектор представлення містить основні ознаки, які описують вхідні значення, і необхідні для максимально точного відтворення даних. Наприклад, навчивши автокодувальник на великому наборі зображень обличь, можна бачити, що після стиснення та декодування обличчя мають більш низьку деталізацію, але основні ознаки зберігаються. Ця властивість використовується при виявленні аномалій. При навчанні автокодувальника стиснене представлення відповідатиме неаномальним даним, бо їх в наборі даних набагато більше. Таким чином, для виявлення аномалій можна орієнтуватися на величину міри відмінності (наприклад, середньоквадратичної помилки) вхідного прикладу від його відтвореного автокодувальником представлення, і якщо ця величина більше певного значення (наприклад, набагато більша за типові значення помилки відтворення в наборі даних), вважати приклад аномальним.

Зображена на рис. 4.2 архітектура є досить загальною, і як кодувальник чи декодувальник можуть використовуватися будь-які моделі. Наприклад, бувають згорткові автокодувальники, які використовують типові для CNN шари. Також *кодувальником може бути запропонована у цьому розділі архітектура моделі для отримання представлень відео*.

## 4.6 Висновки

Отже, було запропоновано метод класифікації відео, що базується на архітектурі, яка використовує згорткові нейронні мережі для визначення вектору представлення окремих кадрів або сегментів відео, послідовність яких обробляється рекурентною нейронною мережею, яка продукує вектор представлення відео. Це багатопотока архітектура, у якій беруться до уваги три потоки: просторовий (послідовність кадрів), часовий (оптичні потоки), звуковий (аудіодоріжка відео). Означена архітектура із CNN і RNN застосовується до кожного потоку, і окрема модель синтезу обчислює фінальний результат класифікації на основі векторів представлень кожного потоку.

Цей підхід поглиблює використання ідеї вивчення представлень, і дозволяє застосовувати запропонований метод для отримання компактного представлення відео, що характеризує різні його ознаки, і може подаватися на вхід іншій моделі, яка, наприклад, вирішує більш високорівневу задачу та може об’єднувати це представлення з іншими даними, або може вирішувати інші задачі на базі цього стисненого опису. Як одне із таких застосувань, було запропоновано використання методу для виявлення аномалій.

Також було запропоновано деякі аспекти вдосконалення методу, такі як доповнення даних (розширення кількості наявних даних, метод регуляризації, більш точні передбачення при доповненні тестових прикладів), змагальне тренування (аби підвищити стійкість моделі змагальних прикладів), використання капсульних мереж (для усунення недоліків CNN і як потенційну заміну явного обчислення оптичного потоку), адаптивна вибірка (аби сегменти відео із більшою кількістю інформації, тобто більш активним рухом, були представлені й більшим обсягом тренувальних прикладів), застосування механізму уваги (для інтерпретації передбачень моделі).

# 5 ПРАКТИЧНІ АСПЕКТИ РЕАЛІЗАЦІЇ ТА НАВЧАННЯ РОЗРОБЛЕНОЇ МОДЕЛІ

## 5.1 Програмне забезпечення

Моделі машинного навчання зручно розробляти на базі бібліотек автоматичного диференціювання. Цей метод маніпулює блоками програм, і обчислює похідну для кожного елементу. Ці елементи є послідовністю арифметичних операцій і елементарних функцій. Диференціатор застосовує правило диференціювання складної функції до цих елементів, і похідні можуть бути обчислені автоматично з достатньою точністю, використовуючи лише на невелике константне число більше арифметичних операцій, ніж вихідна програма. Прикладами таких бібліотек є TensorFlow і PyTorch, які застосовують reverse-mode automatic differentiation [18].

**TensorFlow** [50] – бібліотека для програмування потоків даних із відкритим вихідним кодом, або бібліотека символьної математики, що підтримує автоматичне диференціювання. Розроблена Google, перша версія була опублікована восени 2015 року. Написана на C++, має інтерфейс мовою Python та іншими.

Розробка програм на TensorFlow полягає в визначенні *статичного графу* обчислень, які виконуються після закінчення конструювання графу. Заданий за допомогою API граф транслюється у внутрішнє ефективне представлення. Одним із недоліків такого підходу є складність налагодження програм, оскільки неможливо поставити точку зупинки в будь-якому місці графа та переглянути проміжні значення.

Альтернативою TensorFlow є **PyTorch** [51] – інша бібліотека машинного навчання, розроблена Facebook, яка підтримує автоматичне диференціювання. Відмінністю від TensorFlow є те, що PyTorch використовує *динамічні графи*, які конструюються «на ходу» і дозволяють пришвидшити розробку, усуваючи недоліки TensorFlow. Також варто відмітити, що таку ж функціональність надає розширення TensorFlow, що називається Eager Execution, але його використання дещо знижує продуктивність обчислень (порівняно із статичними графами), що притаманно і

PyTorch, оскільки оптимізації обчислень зазвичай ґрунтуються на знанні повного ланцюжка обчислень (наприклад, при JIT-компіляції).

Найбільш популярною мовою програмування до розробки моделей машинного навчання є **Python** [52] через наявність великої кількості спеціалізованих бібліотек. Крім TensorFlow і PyTorch, наявні також бібліотеки для передобробки даних, візуалізації тощо.

Ще однією корисною бібліотекою є **Keras** [53], яка є високорівневим API (Application Programming Interface – API; прикладний програмний інтерфейс) для нейронних мереж. Вона містить реалізації відомих архітектур нейромереж, а також дозволяє конструювати власні за допомогою типових елементів, таких як згорткові та агрегувальні шари. Keras виконує обчислювальні операції за допомогою іншої бібліотеки для загальних обчислень із підтримкою автоматичного диференціювання, однією із яких є TensorFlow. Таким чином, можна сказати, що за допомогою високорівневих операцій Keras конструюється граф обчислень TensorFlow.

## 5.2 Апаратне забезпечення

Сучасний розвиток і успіх глибинного навчання великою мірою завдячує наявності апаратного забезпечення, яке дозволяє відносно швидко виконувати необхідну для глибоких нейромереж кількість операцій. Таким апаратним забезпеченням є відеокарти (графічні процесори). Вони дозволяють виконувати велику кількість відносно простих операцій паралельно, оскільки типові сучасні графічні процесори мають декілька тисяч обчислювальних ядер. Відеокарти були спочатку розроблені та оптимізовані для обробки зображень, яка ґрунтується на операціях лінійної алгебри над великими блоками даних. Машинне навчання, моделі якого теж переважно побудовані на таких операціях, може значним чином застосовувати потенціал GPU. Використання GPU для різних цілей, пов’язаних не лише із обробкою зображень, набуло назви GPGPU (General Purpose Computing on GPU).

Двома основними виробниками відеокарт наразі є AMD і NVIDIA. Проте сучасні бібліотеки здебільшого підтримують лише NVIDIA, бо від цього виробника наявне програмне забезпечення (CUDA, а також cuDNN), яке значно прискорює обчислення.

TensorFlow і PyTorch підтримують NVIDIA GPU, а також дозволяють виконувати обчислення на центральному процесорі. Однак, тренування моделей на CPU потребує як мінімум на порядок більше часу, тому навчати моделі глибокого навчання за адекватний час можна лише на GPU.

Також набувають поширення спеціалізовані акселератори для машинного навчання (нейронних мереж), прикладом якого є Google TPU (tensor processing unit). TPU наразі доступні лише для оренди в дата-центрах Google, а отже не можуть бути придбані або використовуватись у дата-центрах інших компаній. Таким чином, GPU наразі все одно залишаються найбільш доступним обчислювальним процесором, який надає адекватну продуктивність для застосувань, пов’язаних із машинним навчанням.

## 5.3 Розподілене навчання моделей

Оскільки наявні набори даних є досить великими, а моделі – глибокими та мають значну кількість параметрів, можна використовувати розподілене навчання на декількох (графічних) процесорах.

Два основні підходи до реалізації розподіленого навчання – **паралелізм моделі** та **паралелізм даних**.

У підході **паралелізму даних** [54] пакет (mini-batch) розподіляється по всім наявним GPU так, щоб кожна відеокарта отримала однакову кількість тренувальних прикладів. Далі кожен обчислювальний процесор обчислює градієнт функції втрат після проходження свого пакету прикладів через модель. Потім GPU обмінюються значеннями градієнтів параметрів, агрегують (усереднюють) їх і оновлюють параметри відповідно до отриманих значень. Таким чином, кожен процесор містить

модель цілком, і всі процесори виконують однакові обчислення, але з різними даними.

Проблемою цього підходу є те, що під час зворотного проходу потрібно передати значення градієнтів усіх параметрів усім іншим GPU. Якщо декілька GPU розташовані на одному фізичному сервері, то цей обмін відбуватиметься швидше, оскільки він здійснюватиметься через шину PCI Express. Також існують швидші інтерфейси, такі як NVIDIA NVLink. У випадку кластеру з декількох фізичних серверів, виникає необхідність передавати ці дані через мережу, що набагато повільніше; також цей підхід погано масштабується з ростом кластеру, оскільки мережа це спільний ресурс, і пропускної здатності може не вистачати на комунікацію всіх GPU одночасно.

Дещо знизити вартість обміну даними можна, зменшивши кількість параметрів, градієнти яких потрібно пересилати (у згорткових мережах це можна зробити, наприклад, за допомогою агрегувального або згорткового шару). Також можна використовувати більш вимогливий до кількості обчислень алгоритм оптимізації, такий як RMSProp; тобто витрачатиметься стільки ж часу на пересилання градієнтів, але більше часу піде на обчислення, таким чином збільшуючи завантаження GPU. Також можна дещо прискорити навчання, якщо асинхронно починати обчислення для наступного кроку поки ще пересилаються градієнти з попереднього.

Також проблемним є використання замалого розміру пакету на кожному із процесорів (після розділення). У алгоритмі стохастичного градієнтного спуску розмір пакету обирається так, щоб, маючи декілька різних прикладів у пакеті, рух параметрів у бік зменшення помилки (відповідно до градієнту) також покращував загальну точність моделі (оскільки вибірка прикладів для пакету певним чином характеризує загальний набір даних). Якщо розмір пакету замалий, то він гірше характеризуватиме набір даних, а відповідно процес навчання буде менш стабільним, бо в такому пакеті крок оптимізації може не прямувати в бік локального оптимуму. Але збільшення розміру пакету понад певної кількості дає все менший ефект, а, відповідно, отримуємо майже такий самий результат із більшою кількістю обчислень. Загалом розмір пакету вибирається приблизно в межах 16-128. Але варто відмітити, що розмір

спільної пам’яті у GPU дуже малий (кілька кілобайт); часто реалізації в бібліотеках лінійної алгебри використовують блоки 64x128; тому, якщо розмір пакету менше за 64, зменшується ефективність використання цієї пам’яті. Також, якщо розмір пакету не ділиться на 32, це теж призводить до зменшення ефективності, оскільки потоки стартують блоками по 32 потоки. Таким чином, використання розміру пакету менше за 64 призводить до нижчої ефективності. Тому якщо, наприклад, вибирати пакет розміром 128, і ділити його на 8 GPU, спостерігатиметься значна втрата продуктивності на кожному GPU.

Перевагою паралелізму даних є простота реалізації. На практиці, цей підхід добре працює для згорткових нейронних мереж. Також він ефективний для рекурентних нейромереж, які мають менше параметрів, і обчислювально складні градієнтні кроки.

Альтернативним підходом є **паралелізм моделі** [55]. У ньому модель розподіляється по декільком GPU, і кожна її частина тренується на тих самих даних. Таким чином, кожен процесор працює із частиною моделі, а не частиною даних. Один із підходів до реалізації поділу моделі є розподіл параметрів.

У цьому підході синхронізації виконуються частіше, але вимагають передачі меншої кількості даних. Загалом, це призводить до значно меншого використання пропускної здатності шини або мережі між GPU. Також варто відмітити, що у цьому підході процес навчання повинен дочекатися закінчення процесу синхронізації, бо кожен процесор має лише часткову інформацію, тому не може продовжувати, поки синхронізацію не завершено.

Головною перевагою паралелізму моделей є можливість використання дуже великих нейромереж, параметри яких не вміщаються у пам’ять однієї відеокарти. Також паралелізм моделей краще масштабується, тому доцільно його застосовувати для великих мереж (а паралелізм даних, відповідно, для малих).

Ці два підходи можна комбінувати. У [56] було запропоновано підхід до тренування згорткових нейромереж, який використовує паралелізм даних для згорткових шарів, і паралелізм моделі для повнозв’язних шарів.

Також у контексті розгляду розподіленого навчання варто згадати методи розпаралелювання стохастичного градієнтного спуску. Ці методи є дієвими, коли декілька процесорів мають спільну пам’ять. Наприклад, для потоків (ядер) CPU спільною пам’яттю є оперативна пам’ять; для блоків ядер GPU також доступна спільна пам’ять.

Простим способом (**паралельний SGD із замками**) є додавання блокування на кроці оновлення параметрів: кожен потік захоплює замок, що захищає стан параметрів; читає поточні значення; виконує оновлення (градієнтний крок), і вивільнює замок. Але на практиці часто захоплення замка займає більше часу, ніж саме оновлення, тому цей підхід неефективний.

Одна із модифікація паралельного градієнтного спуску – підхід **HOGWILD! (асинхронний SGD)** [13]. У ньому просто прибрані замки. Таким чином, різні потоки можуть перезаписувати значення один одного (при одночасному записі), а також обчислювати градієнти для застарілого значення стану. Як було доведено експериментально, це не призводить до зменшення математичної ефективності алгоритму, але при цьому досягається значне прискорення.

## 5.4 Висновки

Таким чином, доцільно розробляти моделі машинного навчання за допомогою бібліотек автоматичного диференціювання, таких як TensorFlow і PyTorch, які дозволяють істотно спростити код, а також надають значний набір типових рішень. Ще однією корисною бібліотекою є Keras, який надає більш високорівневий інтерфейс, і дозволяє швидко експериментувати із різноманітними архітектурами нейронних мереж.

Тренування моделей машинного навчання вимагає використання GPU, що дозволяють паралельно виконувати велику кількість операцій лінійної алгебри, із яких переважно й складається більшість обчислень у моделях машинного навчання. Тренування на CPU сучасних моделей є занадто повільним через значну кількість

параметрів у глибоких нейронних мережах. Наразі бібліотеки машинного навчання підтримують лише відеокарти від NVIDIA.

Існує два основних підходи до розподіленого навчання – паралелізм даних і паралелізм моделі. Перший є більш простим у реалізації, і непогано працює для відносно невеликих моделей. Другий є більш складним, але краще масштабується зі збільшенням розміру кластеру, а також дозволяє тренувати моделі, які мають забагато параметрів, щоб їх можна було розмістити в пам’яті однієї відеокарти. Також можливо застосовувати алгоритми паралельного стохастичного градієнтного спуску, які працюють зі спільною пам’яттю та виконують обчислення у декілька потоків на CPU або GPU.

# 6 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЗАПРОПОНОВАНОГО МЕТОДУ

## 6.1 Набір даних

Часто для тестування моделей класифікації відео використовують набори даних, що зображають людські дії [30, 34, 36]. Для експериментів будемо використовувати набір даних UCF101 [30], який містить відеокліпи, які зображають записи людських дій із 101 класу. Цей набір даних має значну різноманітність з точки зору варіацій руху камери, вигляду та позицій об’єктів, масштабу, точки огляду, фону, умов освітлення, та є досить складним для розпізнавання. Наявні класи розділені на 5 груп: взаємодія людини з об’єктом, рух тілом, взаємодія між людьми, гра на музичних інструментах і спортивні активності. UCF101 містить 13320 відео із розподільчою здатністю 320x240 та частотою 25 кадрів на секунду. Середня довжина кліпу – близько 7 секунд, мінімальна – 1 секунда, максимальна – 71 секунда. Загальна довжина відео – 1600 хвилин (приблизно 27 годин). Відповідно загальна кількість кадрів складає 2.4 мільйони. Аудіо доріжка наявна для 51 класу. Цей набір даних містить 3 варіанти поділу на тренувальну та тестову вибірки; надалі навчатимемо моделі та повідомлятимемо результати на першому з них.

Це порівняно невеликий набір даних, який, тим не менш, дозволяє перевірити роботу складних моделей за прийнятний час із результатами, котрі можуть узагальнюватися на більші набори даних.

## 6.2 Вибір оптимальної архітектури рекурентної нейронної мережі

Вже було описано вибрані архітектури компонентів запропонованого багатопотокового методу, але конкретно найкращу архітектуру RNN доцільно визначити експериментальним шляхом.

Для її вибору різні моделі тренувалися на базі LSTM із GRU, однонапрямлені та двонапрямлені, із одним або двома шарами, і різною розмірністю прихованого

стану комірки (256, 512, 1024, 2048); також перевірялося, чи можна покращити точність класифікації за допомогою використання механізму уваги. *Найкраща архітектура вибиралася відповідно до точності класифікації при тренуванні просторового потоку.*

Як вже зазначалося, як згорткову нейромережу було вибрано InceptionResNetV2. Оскільки експериментальним чином було виявлено, що тренування цієї моделі не покращує точність, використовувалася модель із параметрами, натренованими на ImageNet.

**Передобробка.** Стандартною розмірністю вхідного тензора для InceptionResNetV2 є 299x299x3, тому кадри були перемасштабовані до цього розміру. Також було зменшено частоту відео шляхом вибірки 5 кадрів на секунду із постійним кроком. Оскільки довжина часового виміру вхідного тензора RNN має бути фіксована, було вибрано 180 як максимальну довжину послідовності кадрів. Відповідно бралися до уваги лише перші 36 секунд відео (є лише одне відео у наборі даних, яке є довшим). Якщо відео коротше, то порожні місця в тензорі заповнювалися нулями. Розмірність вхідного тензора RNN дорівнює 1536x180, оскільки InceptionResNetV2 повертає 1536-вимірний вектор представлення (з глобального усереднювального агрегувального шару, розташованого перед класифікаційним шаром).

**Технічні параметри.** Метод було реалізовано за допомогою бібліотек TensorFlow версії 1.7 та Keras версії 2.1. Лістинг важливих частин програми наведено в додатку. Тренування проводилося на NVIDIA GTX 1080, яка має 8 Гб відеопам’яті.

**Параметри навчання.** Як оптимізатор використовувався SGD із моментом Нестерова (тобто SGD), оскільки адаптивні градієнтні методи (Adam, RMSProp тощо) давали гірші результати, що також відповідає опублікованим спостереженням [57]. Навчання продовжували допоки зменшувалося значення функції втрат на тестовому наборі даних; якщо протягом 10 епох (епоха – повний прохід по всім тренувальним даним під час навчання моделі) це значення не зменшувалося, тренування припинялося. Початкове значення темпу навчання було вибрано рівним $10^{-2}$; якщо протягом 5 епох значення функції втрат не зменшувалося, знижували темп навчання

в 10 разів (до досягнення мінімального темпу навчання, вибраного рівним $10^{-6}$). Розмір пакету дорівнює 32. У моделях із лінійним SVM коефіцієнт $L_2$ регуляризації є рівним $10^{-2}$. Використовувалося проріджування (dropout) із коефіцієнтом 0.5 на вхідному шарі, між моделлю RNN і повнозв'язним класифікаційним шаром, а також між шарами LSTM/GRU.

Різні варіації архітектур було натреновано із використанням класифікаційного шару із активацією softmax. Найкращу за точністю архітектуру також було протестовано із лінійним SVM (тобто hinge loss, лінійна активація та $L_2$ регуляризація) на останньому шарі замість softmax; також було перевірено точність цієї архітектури із використанням шару уваги, застосованого до вхідного тензора.

Результати експериментів наведено в табл. 6.1. Жирним шрифтом виділено найкращі результати. Назви моделі з приставкою «S.» позначають двошарові архітектури (stacked). Префікс «Bi» означає використання двонапрямлених RNN (bidirectional).

Таблиця 6.1 – Параметри моделей і отримані точності (top-1, top-5) класифікації

| **Модель** | **Розмірність прихованого стану комірки** | **Кількість параметрів** | **Top-1, %** | **Top-5, %** |
|---|---|---|---|---|
| LSTM | 256 | 1,863,013 | 76.24 | 93.49 |
| LSTM | 512 | 4,250,213 | 77.09 | 93.09 |
| LSTM | 1024 | 10,597,477 | 77.07 | 93.62 |
| LSTM | 2048 | 29,583,461 | 76.22 | 93.62 |
| GRU | 512 | 3,200,613 | 77.83 | 93.88 |
| GRU | 1024 | 7,973,989 | 77.81 | 93.17 |
| BiLSTM | 512 | 8,500,325 | 77.36 | 93.41 |
| BiLSTM | 1024 | 21,194,853 | 75.93 | 93.49 |
| BiGRU | 512 | 6,401,125 | 77.33 | 93.46 |
| BiGRU | 1024 | 15,947,877 | 77.52 | 93.41 |
| S. LSTM | 512 | 6,351,461 | 74.71 | 92.80 |
| S. LSTM | 1024 | 18,994,277 | 76.80 | **94.09** |
| S. GRU | 512 | 4,776,549 | 77.09 | 93.03 |
| S. GRU | 1024 | 14,271,589 | 77.22 | 93.25 |
| S. BiGRU | 512 | 11,125,861 | **78.34** | 93.51 |
| S. BiGRU | 1024 | 34,834,533 | 77.15 | 93.17 |
| S. BiLSTM | 512 | 14,799,973 | 74.50 | 92.77 |
| S. BiGRU + attention | 512 | 11,158,441 | 75.03 | 92.89 |
| S. BiGRU + SVM | 512 | 11,125,861 | **79.40** | 90.84 |

Загалом можна бачити, що оптимальними розмірностями прихованого стану комірки LSTM або GRU для цієї задачі є 512 або 1024. Двошарові архітектури працюють краще на базі GRU, порівняно з LSTM. Також моделі RNN із комірками GRU зазвичай давали кращі результати, імовірно за рахунок того, що GRU мають менше параметрів, а відповідно менше схильні до перенавчання. Найкращу точність моделі з softmax активацією (78.34%) було отримано моделлю із двома двонапрямленими шарами GRU, комірки яких мають розмірність прихованого стану, рівну 512. Використання SVM замість softmax дало подальше покращення точності класифікації до 79.40%. Використання шару уваги суттєво знизило точність, тому його варто застосовувати лише в разі необхідності інтерпретувати модель. У підсумку можна сказати, що **найкращою архітектурою RNN в цьому експерименті є модель із двома двонапрямленими шарами GRU із розмірністю прихованого стану комірки, що дорівнює 512.**

## 6.3 Розподілене навчання

Було проведено експеримент для підтвердження можливості розподіленого навчання розробленої моделі та вимірювання його ефективності. У цьому експерименті на просторовому потоці тренувалася рекурентна нейромережа із архітектурою, вибраною у попередньому підрозділі (два двонапрямлених шари GRU, із softmax активацією). Параметри навчання та передобробки є такими самими.

Експеримент проводився у хмарі Amazon EC2. Для навчання із однією GPU використовувався сервер p2.xlarge, що має 1 відеокарту NVIDIA K80 із 12 Гб відеопам’яті. Для розподіленого навчання застосовувався сервер p2.8xlarge, що містить 8 таких відеокарт. Тренувальні дані зчитувалися з SSD-дисків (Amazon EBS) із заявленою продуктивністю у 450 операцій читання/запису на секунду (IOPS).

Розмір пакету на сервер з 1 GPU дорівнював 32. З 8 GPU – 256 (відповідно теж по 32 на відеокарту). Також на сервері з 8 GPU замість 1 потоку зчитування даних із диску використовувалося 8. Розподілене навчання було виконано за допомогою можливостей Keras, що реалізовує *паралелізм даних*.

Результати наведено у табл. 6.2.

Таблиця 6.2 – Результати розподіленого навчання

| | Кількість GPU | |
|---|---|---|
| | 1 | 8 |
| Середній час на епоху, с | 143.4 | 43.6 |
| Час до зупинки тренування, хв | 38.25 | 26.25 |
| Час до досягнення найкращої точності, хв | 38.25 | 22.5 |
| Найкраща точність, top-1 % | 78.28 | 76.98 |

Також далі наведемо графіки (рис. 6.1, 6.2), що показують процес навчання. «acc» (accuracy) позначає точність класифікації на тренувальній вибірці, «loss» – значення функції втрат на тренувальному наборі, «val_acc» (validation accuracy) – точність на тестовій вибірці, «val_loss» (validation loss) – значення функції втрат на тестовому наборі. На горизонтальній осі зображений номер епохи.

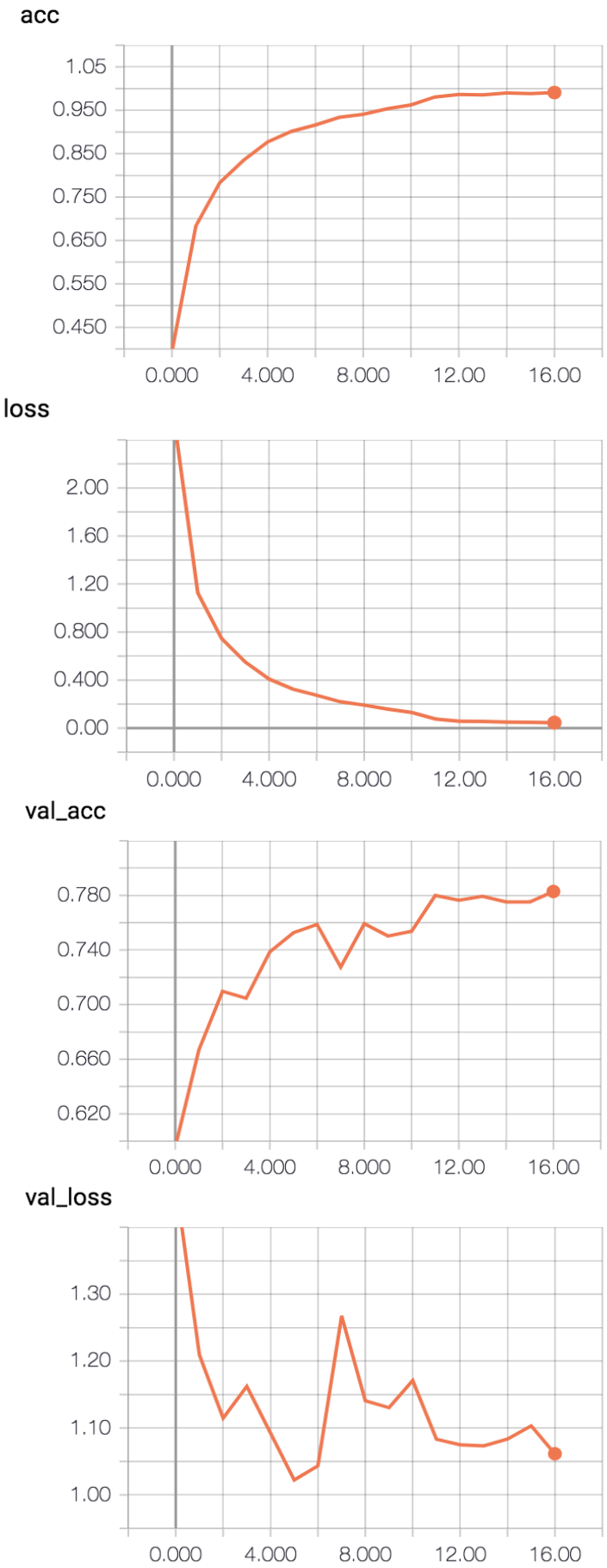



Рисунок 6.1 – Графіки процесу навчання при тренуванні на 1 GPU

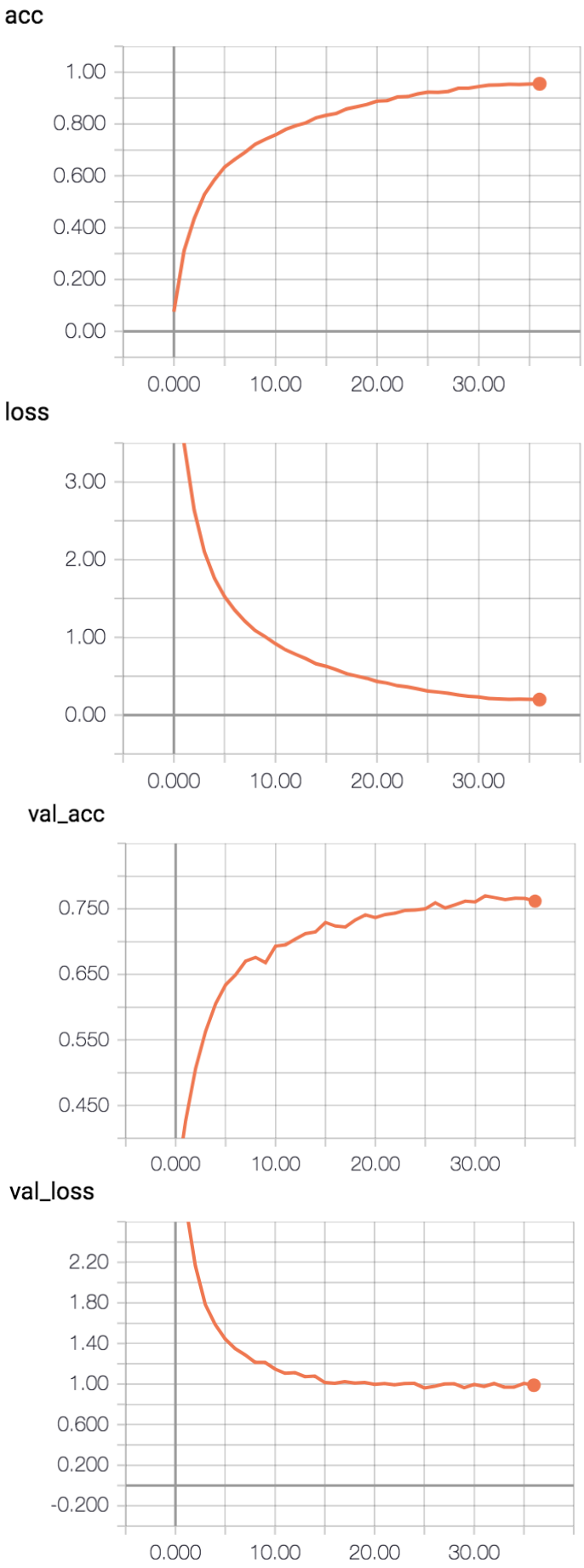



Рисунок 6.2 – Графіки процесу навчання при тренуванні на 8 GPU

Таким чином, спостерігається відоме явище, коли розподілене навчання дає дещо меншу точність через більшу варіативність у стохастичному процесі навчання. Також можна бачити, що використання більших пакетів дає й більш стабільне зменшення значення функції втрат. Загалом, безперечно, використання декількох GPU дає прискорення, але час навчання зменшується далеко не лінійно зі збільшенням кількості відеокарт. Також вимірювання ефективності розподіленого навчання є складним через велику кількість факторів, що безпосередньо впливають на швидкість, а саме продуктивність зчитування з диску та оперативної пам'яті, швидкість процесору тощо. Таким чином, найкращим рішенням є вимірювання ефективності розподіленого навчання для конкретного набору даних і моделі з метою визначення економічної доцільності його використання.

## 6.4 Порівняльний аналіз

Таким чином, було визначено всі компоненти запропонованого методу. Далі було натреновано всі потоки, як це було описано у підрозділі 4.3, виміряно результати класифікації запропонованим методом та здійснено порівняльний аналіз із опублікованими результатами.

Тренування RNN виконувалося з такими самими параметрами, як описано у підрозділі 6.2. CNN для оптичного потоку навчалася за допомогою оптимізатора Adam, оскільки ця мережа тренується без попередньої ініціалізації тренуванням на іншому наборі даних, і більш швидка збіжність Adam у цьому випадку є значною перевагою.

Результати порівняльного аналізу наведено у табл. 6.3.

Таблиця 6.3 – Результати порівняння запропонованого методу з існуючими рішеннями

| | Точність на UCF101, % | Тип моделі | Потоки | Попереднє тренування |
|---|---|---|---|---|
| **Запропонований метод** | 93.1 | 2D CNN + RNN (GRU) | Просторовий, часовий, звуковий | Просторова 2D CNN на ImageNet |

| | | | | |
|---|---|---|---|---|
| Двопотокова архітектура [29] | 88.0 | 2D CNN | Просторовий, часовий | Просторова 2D CNN на ImageNet |
| Багатопотокова архітектура [32] | 92.6 | 2D CNN + RNN (LSTM) | Просторовий, часовий, звуковий | Просторова 2D CNN на ImageNet |
| video2vec [41] | 87.5 | 2D CNN + RNN (GRU) | Просторовий, часовий | Просторова 2D CNN на ImageNet |
| C3D [14] | 90.4 | 3D CNN | Просторовий, часовий | I380K (приватний) + Sports-1M (публічний) |
| I3D [35] | 98.0 | 3D CNN | Просторовий, часовий | ImageNet + Kinetics |
| ResNeXt-101 (64f) [37] | 94.5 | 3D CNN | Просторовий | Kinetics |

Таким чином, можемо бачити, що запропонований метод має кращу точність, ніж існуючі рішення, засновані на архітектурах із двовимірними згортковими та рекурентними нейронними мережами. Набагато кращу точність було отримано порівняно з video2vec [41], яка, як і розроблена модель, обчислює передбачення на базі векторів представлень, а не агрегацією результатів класифікації окремих потоків. Деякі 3D CNN (I3D [35] і ResNeXt-101 (64f) [37]) мають кращі результати, але це набагато складніші моделі, а головне – вони були попередньо натреновані на дуже великому наборі даних (Kinetics).

## 6.5 Висновки

Отже, було описано параметри набору даних UCF101, на якому тестувалося розроблене рішення, та виконане порівняння із існуючими; зазначено параметри експериментів, а також програмне та апаратне забезпечення, які застосовувалися.

Експериментальним чином було **виявлено оптимальну для цієї задачі архітектуру рекурентної нейромережі. Нею виявилися два двонапрямлених шари GRU із розмірністю прихованого стану, що дорівнює 512, і лінійною SVM активацією. Ця архітектура дала точність 79.4% при класифікації просторового потоку.**

Було досліджено застосування розподіленого навчання, саме підходу паралелізму даних. Спостерігалося пришвидшення процесу навчання, але, можливо недостатнє, аби використання багатьох GPU було насправді доцільним. Це великою мірою залежить від конкретної моделі та набору даних.

Також було проведено експеримент із тренуванням усіх потоків запропонованого рішення і моделі синтезу. **Було отримано точність у 93.1%, що переважає існуючі рішення, засновані на архітектурах із 2D CNN і RNN, а також архітектури з 3D CNN, які тренувалися лише на цьому наборі даних.**

# 7 РОЗРОБКА СТАРТАП-ПРОЕКТУ «СЕРВІС ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ГРАФІЧНИХ ДАНИХ»

Розроблена технологія може стати основою стартап-проекту, реалізованого у вигляді інтернет-сервісу, що надаватиме клієнтам послуги інтелектуального аналізу графічних даних. Цей розділ присвячений аналізу різних аспектів розробки та функціонування стартап-проекту, та його економічному обґрунтуванню.

Сервіс даватиме можливість індивідуальним користувачам і організаціям надавати свої графічні дані (зображення чи відео) і отримувати результати їх автоматичного аналізу (за певну платню) за допомогою готових (наперед навчених) моделей. Таким чином вони отримуватимуть корисну інформацію (business insights), використовуючи інтелектуальну власність сервісу.

Також користувачі матимуть змогу надавати описи власних моделей (алгоритмів) і тренувальні дані для їх виконання на інфраструктурі сервісу – в цьому випадку сервіс спрощуватиме їм операційну діяльність, а також надаватиме уніфікований інтерфейс для роботи як з їх власними моделями, так і з готовими (або моделями, розробленими сторонніми організаціями).

## 7.1 Опис ідеї проекту

Таблиця 7.1 – Опис ідеї стартап-проекту

| *Зміст ідеї* | *Напрямки застосування* | *Вигоди для користувача* |
|---|---|---|
| Ідея полягає в створенні сервісу, за допомогою якого організації могли б перетворити свої графічні дані на корисну для бізнесу інформацію | Для індивідуальних користувачів і невеликих компаній із більш типовими задачами | Економія коштів і часу, бо відсутня необхідність розробляти власні рішення |
| | Для клієнтів із нестандартними даними або задачами, або ж спеціальними вимогами до розгортання | Можливість будувати спеціалізовані рішення на підґрунті вже реалізованих базових, які надає сервіс, та економити ресурси за допомогою автоматизації використання обчислювальної інфраструктури |

Таким чином, ключовою ідеєю є надання клієнтам сервісу змоги використовувати інтелектуальну власність сервісу (алгоритми та їхні реалізації) задля аналізу їхніх графічних даних. Серед напрямків застосування основними є два: перший напрямок передбачає використання готових моделей сервісу клієнтами зі стандартними задачами (даними), а другий надає можливість розробляти спеціальні рішення на базі готових. У обох сценаріях використання клієнти заощадживатумить кошти на розробці та розгортанні власних рішень з нуля.

Виділимо основні техніко-економічні характеристики ідеї стартапу, та проведемо порівняльний аналіз із сервісам конкурентів – Google Vision API, Amazon Rekognition, Microsoft Computer Vision API, IBM Watson Visual Recognition. На основі цього аналізу визначимо слабкі, нейтральні та сильні сторони ідеї.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| *№ п/п* | *Техніко-економічні характеристики ідеї* | *(Потенційні) товари/концепції конкурентів* | | | | *W (слабка сторона)* | *N (нейтральна сторона)* | *S (сильна сторона)* |
|---|---|---|---|---|---|---|---|---|
| | | *Мій проект* | *Google* | *Amazon* | *IBM* | | | |
| 1 | Форма виконання | Веб-сервіс | Веб-сервіс | Веб-сервіс | Веб-сервіс | | + | |
| 2 | Собівартість | Середня | Висока | Висока | Висока | | | + |
| 3 | Можливість розгортання в дата-центрі клієнта | Є | Нема | Нема | Нема | | | + |
| 4 | Можливість спеціалізувати рішення під конкретну задачу | Є | Обмежена | Нема | Нема | | | + |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | Відкритість (використання пропрієтарного програмного та апаратного забезпечення) | Розроблений на базі open-source коду та доступного всім апаратного забезпечення (GPU) | Значною мірою базується на закритому коді, частково використовує пропрієтарне апаратне забезпечення | Значною мірою базується на закритому коді | Значною мірою базується на закритому коді, частково використовує пропрієтарне апаратне забезпечення | | | + |
| 6 | Доступ до апаратного забезпечення | Необхідність винаймати потужності в дата-центрах | Простий / дешевий, власні дата-центри, пропрієтарне апаратне забезпечення | Простий / дешевий, власні дата-центри | Простий / дешевий, власні дата-центри, пропрієтарне апаратне забезпечення | – | | |

Отож, сильними сторонами проекту є собівартість розробки (оскільки наявна велика кількість наукових праць у відкритому доступі, частина з них із реалізаціями, тому не потрібно розробляти все з нуля), можливість розгортання в дата-центрі клієнта (користувачі можуть вимагати цього з міркувань захисту даних, але всі сервіси конкурентів передбачають завантаження даних на їх сервери), можливість розробки спеціалізованих рішень (конкуренти дозволяють навчати їх пропрієтарні моделі з користувацькими даними, але не модифікувати свої моделі), відкритість (оскільки більшість моделей будуть засновані на відкритому вихідному коді та опублікованих наукових працях на відміну від конкурентів із закритим кодом їх сервісів; використовуватиметься доступне всім апаратне забезпечення). Слабкою стороною є доступ до обчислювальної інфраструктури, бо в усіх перерахованих конкурентів є великі власні дата-центри, і вони мають контракти на гуртові поставки апаратного забезпечення з виробниками; стартапу ж доведеться винаймати потужності в публічних дата-центрах. Таким чином, з урахуванням зазначених сильних сторін ідеї, даний проект можна вважати конкурентоспроможним.

## 7.2 Технологічний аудит ідеї проекту

У цій секції проведемо аналіз технологій, на основі яких виконуватиметься технічна реалізація проекту, та оцінимо їх доступність.

Таблиця 7.3 – Технологічна здійсненність ідеї проекту

| *№ п/п* | *Ідея проекту* | *Технології її реалізації* | *Наявність технології* | *Доступність технології* |
|---|---|---|---|---|
| 1 | Розробка веб-сервісу | Python, Flask | Наявна | Безкоштовна, доступна |
| 2 | Розробка моделей машинного навчання | Python, TensorFlow, PyTorch | Наявна | Безкоштовна, доступна |
| Обрані технологія реалізації ідеї проекту: для створення сервісу обрана технологія Flask, яка є безкоштовною та доступною; безпосередньо пов'язаний з машинним навчанням код базуватиметься на TensorFlow і PyTorch, які теж є безкоштовними. | | | | |

В результаті проект будуватиметься мовою програмування Python. Розробка веб-сервісу (публічного інтерфейсу проекту) здійснюватиметься за допомогою бібліотеки Flask. Безпосередньо моделі машинного навчання будуть засновані на бібліотеках TensorFlow і PyTorch. Усі зазначені бібліотеки були розроблені як мінімум декілька років тому, активно розробляються та використовуються, а отже є стабільними та добре задокументованими. З цього можна зробити висновок, що перешкод для технічної реалізації проекту нема.

## 7.3 Аналіз ринкових можливостей запуску стартап-проекту

Важливим є аналіз потенційного ринку стартап-проекту для визначення конкурентів (відповідно до цієї інформації важливим є коригування стратегії стартапу для успішного зайняття ринкової ніші), ринкових можливостей (динаміка ринку, рентабельність; для передбачення прибутковості та встановлення цін на сервіси).

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

| *№ п/п* | *Показники стану ринку (найменування)* | *Характеристика* |
|---|---|---|
| 1 | Кількість головних гравців, од | 4 найбільших (Google, Amazon, Microsoft, IBM); до 100 дрібних стартапів |
| 2 | Загальний обсяг продаж, грн/ум.од | $1.35 млрд (ринок Artifical Intelligence (AI) на 2016 рік за даними https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/) |
| 3 | Динаміка ринку (якісна оцінка) | Стрімко зростає (>50% на рік; джерело – те ж, що і в попередньому пункті) |
| 4 | Наявність обмежень для входу (вказати характер обмежень) | Наукоємність, обчислювальні ресурси |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Неусталена стандартизація технологій AI; але наявні вимоги до захисту даних |
| 6 | Середня норма рентабельності в галузі (або по ринку), % | 10-20% для великих компаній (хоча ці дані відносяться до великих компаній взагалі, а не до їх AI підрозділів) |

Таким чином, маємо великий ринок, для якого передбачено стрімке зростання, а також гарні показники рентабельності. Однак, наявні потужні конкуренти зі значними інвестиціями в цю галузь. Входження на ринок утруднюється наукоємністю галузі та значними вимогами до обчислювальних ресурсів для реалізації серйозних проектів. Стандартизація та сертифікація продуктів у галузі перебуває в зародковому стані, але потрібно зважати на вимоги захисту даних.

Охарактеризуємо основні групи потенційних клієнтів та відмінності між ними, аби робити більш інформовані рішення щодо розробки продукту для кожної з цих груп.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту

| *№ п/п* | *Потреба, що формує ринок* | *Цільова аудиторія (цільові сегменти ринку)* | *Відмінності у поведінці різних потенційних цільових груп клієнтів* | *Вимоги споживачів до товару* |
|---|---|---|---|---|
| 1 | Провести інтелектуальний аналіз графічних даних, аби отримати корисні знання з | Аудиторія: компанії та користувачі, які мають дані та хочуть провести їх автоматичний | 1. Для сегменту більш дрібних користувачів більш характерне максимально просте використання готового | Усім користувачам потрібні алгоритми, які дадуть їм достатньо додаткової цінності аби |

| | | | | |
|---|---|---|---|---|
| | «сирих» даних або автоматизувати певні процеси | аналіз, який принесе їм додаткову цінність.<br>Сегменти:<br>1. Індивідуальні користувачі (дослідники) та невеликі підприємства.<br>2. Більші підприємства, які мають великі обсяги даних та готові платити за специфічні рішення під їх конкретні задачі. | продукту без «заточування» під специфічні потреби.<br>2. Більші клієнти можуть потребувати певної стандартизації та QA (особливо, якщо вони автоматизують виробничні процеси) | збільшити прибутковість їхніх компаній. Наприклад, якщо алгоритм автоматичної класифікації зображень розпізнає певні об’єкти на супутниковому знімку краще, ніж людина, або приблизно так само, але використання наших алгоритмів їм дешевше за ручну працю по аналізу їхніх даних. |

У підсумку можна сказати, що є дві групи потенційних користувачів: індивідуальні користувачі та невеликі підприємства, а також більші компанії з великим обсягами та специфічними потребами. Обидві групи об’єднує спільна потреба – отримувати корисну інформацію з «сирих» даних. Для першої групи потрібно зробити акцент на простоту використання та ціну. Специфічні потреби другої групи вимагатимуть додатковий контроль якості та розробку спеціалізованих рішень.

Тепер проаналізуємо фактори загроз, які можуть негативно вплинути на діяльність і результати роботи підприємства, та передбачимо можливу реакцію на виникнення цих загроз.

Таблиця 7.6 – Фактори загроз

| *№ п/п* | *Фактор* | *Зміст загрози* | *Можлива реакція компанії* |
|---|---|---|---|
| 1 | Зростаюча конкуренція | Великі IT-компанію вкладають багато грошей в розвиток AI, розуміючи потенціал галузі. Наукові установи (університети, лабораторії) фокусуються на цих дослідженнях, отримують більше грантів. | Пробувати покращити існуючі технології; винайти нові сфери застосування; залучити компанії, що раніше не застосовували технології AI, до їх використання. |
| 2 | Велика кількість дешевої робочої сили | Дозволяє компаніям використовувати робітників у різноманітних виробничих процесах і аналізі | Намагатися зменшити витрати, аби зменшити ціни. Наприклад, оптимізувати алгоритми для |

| | | | |
|---|---|---|---|
| | | даних, витрачаючи на них в деяких випадках менше, ніж на складні технології і обчислювальні ресурси. | використання меншої кількості обчислювальних ресурсів. Домовитися з вендорами обчислювальних ресурсів про партнерство та кращі умови. |
| 3 | Достатня закритість наукових досягнень, патентування | Незважаючи на наукові публікації, справжній науковий рівень гравців ринку невідомий, бо вони добре захищають свою інтелектуальну власність. Також деякі відомі алгоритми не можна використовувати (або це дорого) через те, що вони запатентовані. | Досліджувати можливості конкурентів; виконувати reverse engineering (в межах закону); досліджувати запантентовані рішення, аби уникнути порушень; проводити власні наукові розробки. |
| 4 | Дорожчання або припинення доступу до обчислювальних ресурсів | Конкуренти мають власні дата-центри, сервісу доведеться винаймати потужності в публічних дата-центрах. До того ж, конкуренти стартапу в галузі AI і є власниками найбільших дата-центрів. У разі, якщо стартап досягне значного успіху та становитиме загрозу їхній ринковій частці, вони можуть підняти ціну на оренди їхніх серверів або взагалі розірвати контракт. | Розробляти інструменти автоматизації розгортання, аби мати можливість швидко перейти до іншого постачальника обчислювальних потужностей; розгортати сервіс у дата-центрах різних компаній, аби зменшити ризик у разі негативних дій однієї або декількох із них; досліджувати можливості використання дата-центрів не від компаній-конкурентів. |
| 5 | Сповільнення зростання галузі | Зараз галузь AI знаходиться на стадії дещо завищених очікувань. Багато задач вирішуються за допомогою підходів машинного навчання, але можливості обмежені, і науковий прогрес у цій галузі бути повільнішим за очікування бізнесу. Це може призвести до сповільнення росту галузі, коли завищені очікування зіштовхуться з реальністю обмежених можливостей. | Робити більш консервативні прогнози щодо майбутньою прибутковості, кількості клієнтів і галузей застосування тощо; залучати клієнтів, які ще не користуються технологіями AI задля того, щоб компенсувати зниження попиту серед тих, хто вже їх використовує; знаходити області застосування такі, які би давали корисні результати незважаючи на обмеження технологій. |

Отже, значну конкуренцію на цьому ринку варто компенсувати технологічністю та інноваційністю; на низьку ціну на ручне виконання тих задач, які вирішує сервіс, варто реагувати здешевленням послуг за допомогою зниження собівартості, якої можна досягнути оптимізацією коду; на патентні ризики та закритість інтелектуальної власності варто відповідати проведенням власних досліджень і створенням власних технологій, а також контролем використання рішень, які можуть бути запатентовані; потрібно також мати альтернативні варіанти на випадок підвищення ціни або припинення доступу до обчислювальних ресурсів

від дата-центрів компаній-конкурентів; виконувати економічне планування з урахуванням можливого сповільнення росту галузі.

Далі проаналізуємо фактори можливостей, аби їх оптимальним чином використовувати на свою користь.

Таблиця 7.7 – Фактори можливостей

| *№ п/п* | *Фактор* | *Зміст можливості* | *Можлива реакція компанії* |
|---|---|---|---|
| 1 | Зростаючий ринок AI | Збільшення попиту на технології штучного інтелекту. | Намагатися захопити більшу частину ринку; проектувати рішення, яке здатне масштабуватися. |
| 2 | Величезна кількість ручної праці, яку можна автоматизувати, та зростаючі соціальні стандарти | Багато категорій даних і виробничих процесів, де алгоритми машинного навчання потенційно можуть замінити людей, що, в свою чергу, зменшить витрати для компаній, які використовують людську працю. | Пропонувати послуги компаніям, які не використовують машинне навчання, шукати можливість покращити їх ефективність за допомогою нашого продукту. |
| 3 | Достатньо високий рівень наукових досягнень в галузі | Наприклад, алгоритми класифікації зображень на даний час мають таку саму, як люди, або кращу точність. Галузь не зовсім нова, тому наявна велика кількість наукових досягнень, на яких можна ґрунтувати продукт. | Використовувати існуючі знання та технології як базу, не винаходячи все з нуля. Шукати шляхи покращення існуючих рішень і нові галузі їх застосування. |
| 4 | Дешевшають обчислювальні ресурси | Саме це і поява нових апаратних засобів (обчислення на відеокартах) дало поштовх дуже значному росту точності моделей машинного навчання в останні роки. | Можливо будувати більш складні алгоритми, які вирішуватимуть задачі краще, але при цьому вартість і час їх обчислення все одно залишатимуться на допустимому рівні. |
| 5 | Сприятливі юридичні та регуляторні умови | Оскільки технології машинного навчання доводять свою ефективність, регулятори схильні довіряти їм. Наприклад, деякі компанії (як Google) вже мають ліцензії на використання автономних авто на дорогах загального призначення. Також законодавці схильні | Можна з меншими витратами впроваджувати рішення через менші витрати на юридичні погодження. |

| | | | |
|---|---|---|---|
| | | дозволяти автоматизувати ручну працю, якщо це зменшить травматичність і смертність на виробництві (що в свою чергу зменшить державні витрати на охорону здоров'я). | |

Отже, серед можливостей у галузі AI можна виділити загальне зростання попиту на такі технології, з урахування якого треба робити фокус на збільшення ринкової долі стартапу; також варто залучати компанії з традиційних галузей, які ще не використовують підходи машинного навчання, до використання сервісу, для максимальної реалізації потенціалу глобального тренду на автоматизацію ручної праці; досліджувати та використовувати наявні технології, аби здешевити розробку сервісу за рахунок використання доступних наукових доробок; будувати більш складні (масштабні) моделі з урахуванням зниження цін на обчислювальні ресурси; менше витрачати коштів на юридичні аспекти через низький поточний рівень зарегульованості галузі.

Тепер здійснімо аналіз конкуренції, аби охарактеризувати її характеристики та зрозуміти вплив на діяльність стартапу та передбачити кроки збільшення конкурентноспроможності.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

| *Особливості конкурентного середовища* | *В чому проявляється дана характеристика* | *Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)* |
|---|---|---|
| 1. Вказати тип конкуренції:<br>- **монополістична** | • Значну частину ринку займають великі компанії, але є й успішні стартапи;<br>• Багато невирішених задач, тому можливо ввійти в галузь за допомогою нових винаходів і послуг;<br>• Технології в багатьох випадках важливіше за ціни. | • Можливо спробувати займати ніші, не зайняті іншими (як то аналіз відео);<br>• За наявності кращих технологій можна встановлювати вищі ціни, аби мати вищу маржу або компенсувати малі обсяги. |

| | | |
|---|---|---|
| 2. За рівнем конкурентної боротьби:<br>- **глобальний** | • На місцевому ринку дуже мало підприємств, що працюють в AI, і майже нема покупців;<br>• Конкуренція науки і технологій завжди глобальна. | • Складніше конкурувати через велику кількість підприємств з усього світу, що працюють в цій галузі;<br>• Аби успішно конкурувати, треба враховувати світові досягнення в AI та залучати іноземних фахівців. |
| 3. За галузевою ознакою<br>- **внутрішньогалузева** | • В основному це конкуренція технологій і цін усередині галузі (Artifical Intelligence, або більш глобально – в IT).<br>• Досить рідко конкуренція буває міжгалузевою, коли, наприклад, за допомогою технологій AI можна замінити технології з інших галузей (скажімо, автоматизувати певний виробничий процес). | • Постачальники з тієї ж галузі, що може становити загрозу.<br>• AI є похідною галуззю IT, тому доводиться конкурувати з великими IT-компаніями, які багато інвестують в R&D. |
| 4. Конкуренція за видами товарів:<br>- **товарно-родова** | • Конкуренція між різними технологіями, які з різною якістю та за різну вартість виконують подібні бізнес-задачі. | • Наявна конкуренція технологій, але задачі в основному визначені, тому в основному конкуренція за вирішення одних і тих же (досить великих) категорій задач. |
| 5. За характером конкурентних переваг<br>- **неціновa** | • На даний час в основному якість технології має більше значення за ціну.<br>• Часто буває, що краща технологія вирішує задачу в багато разів краще за використання тієї ж кількості обчислювальних ресурсів. | • Можна сконцентруватися на вирішенні наукових задач, бо обчислювальні ресурси дешевшають. |
| 6. За інтенсивністю<br>- **не марочна** | • В основному значення мають технології і ціни, і лише потім споживачі звертають увагу на бренд.<br>• Утім, споживачі схильні більше довіряти великим компаніям, які мають більше ресурсів на R&D і можуть створювати більш стабільне програмне забезпечення. | • Можливо конкурувати за технологіями, не звертаючи особливої уваги на бренд. |

З урахуванням поданого аналізу варто відмітити, що виходячи з характеру конкуренції основний акцент варто робити на технологічності, оскільки ціни та бренд наразі мають другорядне значення; через глобальний характер конкуренції варто орієнтуватися на світові наукові досягнення; внутрішньогалузева конкуренція має

негативні ефекти через значний доступ до капіталу великих IT-компаній та той факт, що вони ж і є постачальниками обчислювальних ресурсів.

Далі проаналізуємо конкуренцію в галузі відповідно до п’яти факторів, виділених М. Портером. Зокрема, зазначимо прямих конкурентів, потенційних конкурентів і бар’єри входження на ринок, фактори сили постачальників і споживачів, а також дослідимо загрози від товарів-замінників.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

| *Складові аналізу* | *Прямі конкуренти в галузі* | *Потенційні конкуренти* | *Постачальники* | *Клієнти* | *Товари-замінники* |
|---|---|---|---|---|---|
| | *Навести перелік прямих конкурентів* | *Визначити бар’єри входження в ринок* | *Визначити фактори сили постачальників* | *Визначити фактори сили споживачів* | *Фактори загроз з боку замінників* |
| Висновки: | Прямі конкуренти – сервіси від Google, Amazon, Microsoft, IBM і великої кількості стартапів. Вже є дуже інтенсивна конкуренція на ринку AI між великими компаніями, які намагаються продати свої рішення користувачам інших своїх продуктів і агресивно інвестують в R&D та захоплюють ринок. | Бар’єр входження в ринок досить невисокий через значний рівень опублікованих наукових досягнень і відносно низьку ціну обчислювальних ресурсів. Є невеликі стартапи, які теж входять на ринок. Невідомі строки виходу на ринок подібних сервісів від великих гравців на ринку. | Є велика конкуренція між постачальниками (провайдерами обчислювальних ресурсів), тому вимог з їх боку майже нема. | Клієнтам потрібен достатній рівень запропонованих технологій, аби їх впровадження було економічно доцільним, а також вони мають додаткові потреби (захист інтелектуальної власності, даних). | Товаром-замінником є ручна праця, яка досить дешева в деяких регіонах, а також може використовуватися через зручні сервіси (як Amazon Mechanical Turk, CrowdFlower). |

Отже, бачимо, що основними конкурентами є сервіс від крупних IT-компаній, а також менших стартапів або дослідницьких лабораторій. Вплив постачальників незначний, оскільки є багато провайдерів обчислювальних ресурсів, які однаково

доступні учасникам ринку. Клієнтам важлива технологічність, а наявність якісних алгоритмів також нівелює вплив товару-замінника, яким є ручний аналіз даних.

Тепер визначимо, за якими ключовими факторами запропонований сервіс буде кращим за ті, які пропонують конкуренти.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

| *№ п/п* | *Фактор конкурентоспроможності* | *Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)* |
|---|---|---|
| 1 | Технологічність | Пропонування кращих (більш точних, більш продуктивних) алгоритмів для вирішення задач клієнтів. |
| 2 | Спрощення доступу до технологій | Прибирання бар'єрів і спрощення початку використання технологій AI для клієнтів, що зменшить невизначеність з їх боку та знизить їхні витрати. |
| 3 | Залучення нових клієнтів до галузі | Пропонування наших рішень клієнтам, які ще не використовують машинне навчання, а застосовують більш традиційні методи виробництва і аналізу даних (ручна праця). |
| 4 | Новий продукт – аналіз відео | Пропонувати нове рішення, яке мало представлене на ринку. |
| 5 | Додаткові послуги для великих клієнтів | Спеціалізація рішень під вимоги певних клієнтів, задоволення їх специфічних потреб (скажімо, для data protection compliance – розгортання наших алгоритмів у їх дата-центрах) |

Отож наш проект виділятиметься наявністю моделей машинного навчання з нижчою собівартістю використання (з точки зори обчислювальних ресурсів). Збільшувати ринкову частку пропонується за рахунок спрощення використання наших технологій і розгортання рішень у компаніях (на виробництвах), які поки що не користуються перевагами AI. Іншими перевагами є нова ніша (аналіз відео) та задоволення потреб захисту даних клієнтів.

Наступним кроком проаналізуємо сильні та слабкі сторони стартап-проекту. Окремо виконаємо порівняння з двома основними групами конкурентів – сервісами від великих IT-компаній і маленькими стартапами, які фокусуються саме на галузі AI.

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін сервісів від великих конкурентів – Google Vision API, Amazon Rekognition, Microsoft Computer Vision API, IBM Watson Visual Recognition

| *№ п/п* | *Фактор конкурентоспроможності* | *Бали 1-20* | *Рейтинг товарів-конкурентів у порівнянні з Google, Amazon, Microsoft, IBM* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *–3* | *–2* | *–1* | *0* | *+1* | *+2* | *+3* |
| 1 | Технологічність | 8 | | | ✓ | | | | |
| 2 | Спрощення доступу до технологій | 12 | | | | | | ✓ | |
| 3 | Залучення нових клієнтів до галузі | 15 | | | | | | | ✓ |
| 4 | Новий продукт – аналіз відео | 17 | | | | | | | ✓ |
| 5 | Додаткові послуги для великих клієнтів | 20 | | | | | | | ✓ |

Таблиця 7.12 – Порівняльний аналіз сильних та слабких сторін сервісів від стартапів

| *№ п/п* | *Фактор конкурентоспроможності* | *Бали 1-20* | *Рейтинг товарів-конкурентів у порівнянні з іншими дрібними стартапами* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | *–3* | *–2* | *–1* | *0* | *+1* | *+2* | *+3* |
| 1 | Технологічність | 16 | | | | | | | ✓ |
| 2 | Спрощення доступу до технологій | 12 | | | | | ✓ | | |
| 3 | Залучення нових клієнтів до галузі | 10 | | | | ✓ | | | |
| 4 | Новий продукт – аналіз відео | 16 | | | | | | | ✓ |
| 5 | Додаткові послуги для великих клієнтів | 18 | | | | | | | ✓ |

Таким чином, бачимо, що наш стартап-проект переважає аналоги від дрібних стартапів за технологічністю, але дещо відстає за цим критерієм від AI відділів великих компаній, які мають кращий доступ до кваліфікованих науковців. Сервіси від великих IT-компаній роблять менший фокус на залучення клієнтів із нових галузей, і це є перевагою нашого стартап-проекту, але інші стартапи теж роблять фокус на збільшення своєї ринкової долі за рахунок таких клієнтів. Фактор спрощення використання технологій меншою мірою проявляється порівняно із великими конкурентами, бо вони також роблять акцент на цьому, але в результаті мають менш гнучкі рішення. Пропозиція аналізу нового виду даних (відео) та розгортання моделей

в дата-центрах клієнтів для задоволення потреб захисту є сильними сторонами в порівнянні з обома групами конкурентів.

Тепер виконаємо SWOT-аналіз стартап-проекту для визначення сильних і слабких сторін запропонованого сервісу, а також окреслимо можливості та загрози виходу на ринок.

Таблиця 7.13 – SWOT-аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони:<br>• Фокус на дослідження та впровадження нових підходів, покращення існуючих<br>• Спрощення використання технологій AI для клієнтів<br>• Залучення клієнтів, які ще не використовують машинне навчання, але можуть мати зиск з його використання<br>• Новий продукт – аналіз відео, якого нема в конкурентів<br>• Пропозиція додаткових послуг для клієнтів, спеціалізація рішення під їхні потреби | Слабкі сторони:<br>• Менша кількість ресурсів і нижчий початковий рівень технологій та доступу до ресурсів, аніж у великих IT-компаній<br>• Входження в досить конкурентний ринок Computer Vision<br>• Вартість продукту може виявитись зависокою, аби замінити ручну працю в країнах з дешевою робочою силою<br>• Початкова орієнтація продукту на вирішення загальних задач |
| Можливості:<br>• Зростаючий ринок AI<br>• Величезна кількість ручної праці, яку можна автоматизувати, та зростаючі соціальні стандарти<br>• Достатньо високий рівень наукових досягнень в галузі<br>• Дешевшають обчислювальні ресурси<br>• Сприятливі юридичні та регуляторні умови | Загрози:<br>• Зростаюча конкуренція<br>• Велика кількість дешевої робочої сили<br>• Достатня закритість наукових досягнень, патентування |

Можна зробити висновок, що стартап-проект виходитиме на ринок із досить потужною конкуренцією, що компенсується постійно зростаючим попитом на технології AI через глобальний тренд на автоматизацію ручної праці та здешевлення обчислювальних ресурсів. Значний рівень розвитку сфери аналізу графічних даних є одночасно і позитивним, і негативним фактором, бо з одного боку наявна велика кількість інструментів розробки моделей машинного навчання та наукових праць у

відкритому доступі, а з іншого значна частина рішень конкурентів (особливо великих) базується на закритих розробках.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 7.14 – Альтернативи ринкового впровадження стартап-проекту

| *№ п/п* | *Альтернатива (орієнтовний комплекс заходів) ринкової поведінки* | *Ймовірність отримання ресурсів* | *Строки реалізації* |
|---|---|---|---|
| 1 | Розробка веб-сервісу із простим і добре задокументованим інтерфейсом для загального користування, на основі якого користувачі самостійно виконуватимуть розробку власних специфічних рішень | 80%<br>Висока, оскільки сервіс зможе надавати послуги широкому колу клієнтів, тобто не буде ризиків, пов’язаних із залежністю від конкретного великого клієнта | 9 місяців<br>Цей варіант ринкового впровадження є досить трудомістким, оскільки вимагатиме розробки достатньо загальних рішень, які можуть працювати з великою кількістю задач без додаткового налагодження |
| 2 | Розробка спеціалізованих рішень для одного або декількох клієнтів, які будуть налаштовані під їхні конкретні потреби | 25%<br>Вимагатиме пошуку клієнтів до початку розробки, що складніше зробити, не маючи працюючого продукту. Значні ризики пов’язані з залежністю від декількох великих клієнтів. Складність адаптації спеціалізованих рішень для широкого загалу | 6 місяців<br>Швидше за перший варіант через відсутність необхідності розробляти загальнодоступний веб-сервіс, документувати його та тестувати на великому числі різноманітних задач |

Із означених альтернатив обираємо ту, для якої отримання ресурсів є більш імовірним, тобто №1. Хоча вона трохи довша в реалізації, в результаті буде створено сервіс, яким зможе користуватися набагато більша кількість клієнтів. Маючи працюючи веб-сервіс, що достатньо добре вирішує загальні задачі, як наступний крок стартап також зможе пропонувати клієнтам спеціалізацію під їхні задачі.

## 7.4 Розробка ринкової стратегії проекту

Проаналізуємо цільові групи потенційних споживачів, їх готовність використовувати наш продукт і прогнозований попит, а також оцінимо інтенсивність конкуренції та простоту входу у кожен із сегментів.

Таблиця 7.15 – Вибір цільових груп потенційних споживачів

| *№ п/п* | *Опис профілю цільової групи потенційних клієнтів* | *Готовність споживачів сприйняти продукт* | *Орієнтовний попит в межах цільової групи (сегменту)* | *Інтенсивність конкуренції в сегменті* | *Простота входу у сегмент* |
|---|---|---|---|---|---|
| 1 | Індивідуальні користувачі та невеликі підприємства, яким потрібно просто аналізувати невеликі обсяги даних | Висока, бо вони усвідомлюють потребу у вирішенні задачі та обирають кращого постачальника рішення | Високий, бо користувачі в цілому знайомі з цією індустрією. | Висока, бо великі компанії (Google, Amazon) часто апелюють до простоти використання та масового споживача | Відносно просто, бо не потрібно підлаштовувати продукт під специфічні вимоги |
| 2 | Великі підприємства зі специфічними задачами, які можуть усвідомлювати потребу в подібних технологій або ж не використовувати їх взагалі | 1. Якщо клієнти усвідомлюють потребу в продукті, то вони готові його використовувати, але шукають найкраще рішення.<br>2. Підприємства, що ще не використовують технології машинного навчання, можуть бути не готові до змін у своїй виробничій діяльності, і їм потрібно обґрунтовувати переваги автоматизації. | 1. Високий, бо підприємства шукають найкраще рішення.<br>2. Невисокий, бо великі підприємства часто не дуже гнучкі в перебудові своїх виробничих процесів. | Відносно невисока, бо великі підприємства (Google, Amazon) нечасто проактивно контактують з окремими підприємствами з метою продажу свого рішення або пропозицій по розробці спеціалізованого рішення. | Відносно складно, бо великі клієнти шукають надійного і відомого постачальника, який відповідає усталеним стандартам якості. |
| Які цільові групи обрано: індивідуальні користувачі та невеликі підприємства, великі підприємства. | | | | | |

Таким чином було виділено дві основні цільові групи. Індивідуальних користувачам і невеликим підприємствам притаманний певний досвід використання подібних продуктів, тому вони шукатимуть найбільш технологічне рішення. Для другої групи клієнтів (великі підприємства) характерні значні вимоги до якості продукту та захисту даних, а також складність перебудови своїх підходів для використання нових технологій.

Проаналізуємо альтернативи розвитку проекту з точки зору стратегії охоплення ринку, факторів конкурентноспроможності та виділимо базову стратегію розвитку.

Таблиця 7.16 – Визначення базової стратегії розвитку

| *№ п/п* | *Обрана альтернатива розвитку проекту* | *Стратегія охоплення ринку* | *Ключові конкурентоспроможні позиції відповідно до обраної альтернативи* | *Базова стратегія розвитку** |
|---|---|---|---|---|
| 1 | Розробку масового продукту під широке коло задач | Фокус на інноваційності | Загальний продукт для широкої аудиторії | Стратегія диференціації – створення кращої технології, ніж в конкурентів |
| 2 | Розробка спеціалізованих рішень для великих підприємств | Залучення нових клієнтів в галузь, фокус на перевагах автоматизації. Пропозиція розробки рішення, що найкращим чином вирішує задачу конкретного клієнта | Контрактна розробка спеціалізованих рішень | Стратегія спеціалізації – концентрація на вирішенні проблем окремих клієнтів |

Як вже було зазначено, було обрано альтернативу №1, яка передбачає використання стратегії диференціації, що полягатиме в створенні більше інноваційного продукту, що призначатиметься широкому колу користувачів. Альтернатива №2, яка може бути реалізовано після початкового запуску стартапу, передбачатиме спеціалізацію рішень з метою розробки рішень «під ключ» для клієнтів, які не бажають самостійно реалізовувати свої рішення на базі наших загальнодоступних моделей.

Далі виберемо стратегію конкурентної поведінки з урахуванням таких аспектів, як наявність подібних продуктів на ринку, взаємодію зі споживачами продуктів від конкурентів, та підхід до уподібнення до конкурентів за характеристиками продукту.

Таблиця 7.17 – Визначення базової стратегії конкурентної поведінки

| *№ п/п* | *Чи є проект «першопрохідцем» на ринку?* | *Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?* | *Чи буде компанія копіювати основні характеристики товару конкурента, і які?* | *Стратегія конкурентної поведінки** |
|---|---|---|---|---|
| | Ні. Існують подібні сервіси | 1. Шукати нових споживачів, особливо з областей, які ще не користуються технологіями машинного навчання.<br>2. Забирати існуючих у конкурентів за допомогою створення кращого продукту. | Копіюватиметься перелік вирішуваних задач (скажімо, тегування відео, розпізнавання обличь), вдалі підходи до побудови інтерфейсу. Використовуватимуться загальновідомі в науці методи, які покращуватимуться, за рахунок чого продукт буде інноваційним. | Стратегія виклику лідера: пошук недоліків продуктів конкурентів та їх вирішення, фокус на покращенні технологій, пропозиція нових способів використання продукту (розгортання на інфраструктурі замовлення), спеціалізація під конкретні задачі замовника, залучення нових клієнтів до галузі. |

Отже, слідуватимемо стратегії виклику лідера. Стартап зробить фокус на покращенні результатів подібних сервісів від конкурентів та усуненні їх недоліків, але вирішуватиме в основном той самий набір задач. Також залучатимуться клієнти, що ще не користуються подібними сервісами; іншим же пропонуватиметься краща якість вирішення їхніх задач із нижчою собівартістю.

Зважаючи на описані вимоги клієнтів, а також стратегії розвитку та конкурентної поведінки, визначимо стратегію позиціонування.

Таблиця 7.18 – Визначення стратегії позиціонування

| *№ п/п* | *Вимоги до товару цільової аудиторії* | *Базова стратегія позиціонування* | *Ключові конкурентоспроможні позиції власного стартап-проекту* | *Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)* |
|---|---|---|---|---|
| 1 | Висока якість вирішення широкого кола задач | Позиціонування за показниками якості, технології | Загальний продукт для широкої аудиторії | Інноваційність, простота, ефективність |
| 2 | Спеціалізовані рішення під певні задачі | Позиціонування за конкретною сферою застосування | Контрактна розробка спеціалізованих рішень | Спеціалізованість, compliance, якість |

Отже, призначене для широкого загалу рішення позиціонуватиметься як якісне, технологічне та інноваційне, але в той же час ефективне. Спеціалізовані рішення позиціонуватимуться відповідно до сфери застосування, і при контрактній розробці подібних рішень робитиметься акцент на стандартах якості та задоволенні специфічних потреб, таких як додатковий рівень захисту даних.

## 7.5 Розробка маркетингової програми стартап-проекту

Спочатку визначимо ключові переваги потенційного сервісу перед конкурентами виходячи з потреб клієнтів, а також вигоди, які пропонує товар.

Таблиця 7.19 – Визначення ключових переваг концепції потенційного товару

| *№ п/п* | *Потреба* | *Вигода, яку пропонує товар* | *Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити* |
|---|---|---|---|
| 1 | Виконувати автоматичний аналіз графічних даних | Здешевлення обробки даних, підвищення якості | Простота використання; інноваційні алгоритми, які забезпечують високу точність |
| 2 | Розгортати моделі машинного навчання клієнтів на інфраструктурі підприємства | Скорочення часу і витрат на підтримку інфраструктури | Наявність простого інтерфейсу, інструментів моніторингу |
| 3 | Розгортати рішень сервісу на інфраструктурі клієнта | Compliance у сфері захисту даних для клієнтів, для яких це критично | Наявність такої можливості, автоматизація такого розгортання, захист технологій підприємства від крадіжки інтелектуальної власності клієнтами під час такого розгортання |

Таким чином, стартап пропонуватиме більш дешевий та якісний автономний аналіз графічних даних із простим інтерфейсом, що надаватиме можливість користувачам відслідковувати процеси навчання та використання моделей, а також виділятиметься можливістю розгортання базових або спеціалізованих моделей на потужностях замовника за допомогою інструментів автоматизації.

Наступним кроком розробимо трирівневу маркетингову модель товару, які визначатиме конкретний задум сервісу, його реалізацію, а також супровід.

Таблиця 7.20 – Опис трьох рівнів моделі товару

| *Рівні товару* | *Сутність та складові* | | |
|---|---|---|---|
| I. Товар за задумом | Аналіз графічних даних. Автоматизує ручну працю, знижуючи витрати, підвищуючи швидкість і якість аналізу. | | |
| II. Товар у реальному виконанні | Властивості/характеристики: | М/Нм | Вр/Тх/Тл/Е/Ор |
| | 1. Простий для використання інтерфейс. | Нм | Тх |
| | 2. Висока продуктивність (швидкість) аналізу. | Нм | Тх |
| | 3. Висока точність аналізу. | Нм | Тх |
| | Якість: стандартні підходи до QA програмного забезпечення, вибіркова ручна валідація результатів. | | |
| | Пакування: оформлення сайту продукту. | | |
| | Марка: сервіс інтелектуального аналізу графічних даних. | | |
| III. Товар із підкріпленням | До продажу: пропозиція специфічних моделей розгортання. | | |
| | Після продажу: підтримка, спеціалізація під конкретні задачі. | | |
| За рахунок чого потенційний товар буде захищено від копіювання: технічний захист коду; ліцензія; патентування технічних рішень. | | | |

Таким чином, метою сервісу є автоматизація аналізу графічних даних. При розробці акцент робитиметься на ефективність, точність розпізнавання та простоту використання. Доступ і продаж послуг здійснюватиметься через сайт. Серед додаткових переваг для споживача можна виділити пропозицію технічної підтримки, а також розширення співробітництва через розробку спеціальних рішень. Захист

інтелектуальної власності реалізовуватиметься через юридичні інструменти (ліцензування, патентування), а також технічні.

Далі проаналізуємо ціни на товари-аналоги та рівні доходів клієнтів, і встановимо межі цін на послуги виходячи з них.

Таблиця 7.21 – Визначення меж встановлення ціни

| *№ п/п* | *Рівень цін на товари-замінники* | *Рівень цін на товари-аналоги* | *Рівень доходів цільової групи споживачів* | *Верхня та нижня межі встановлення ціни на товар/послугу* |
|---|---|---|---|---|
| 1 | Класифікація зображень.<br>Ручна праця: $10/годину (Amazon Mechanical Turk).<br>15 секунд на зображення => 240 зображень за $10 => ~$42/1000 зображень | • $1-$1.5 (залежно від обсягу) за 1000 зображень (Google)<br>• $0.4-$1 (залежно від обсягу) за 1000 зображень (Amazon) | • $2000-$15000/місяць для індивідуальних користувачів/дослідників<br>• Для підприємств: залежно від розміру підприємства та бюджету. | $1-2/1000 зображень |
| 2 | Класифікація відео.<br>Ручна праця: $10/годину (Amazon Mechanical Turk) => $0.167/хвилину<br>20 секунд на аналіз 1 хвилини відео => $0.223/хвилину | $0.1/хвилину (Google, Amazon) | | $0.1-$0.2/хвилину |

Таким чином, було встановлення межі встановлення цін на класифікацію зображень і відео. Ціни є конкурентними порівняно із виконанням подібних задач вручну, та на майже такому ж рівні, як у сервісів-конкурентів. Верхні межі трохи вище від тих, що пропонують конкуренти з огляду на кращі моделі машинного навчання та додаткові послуги.

Наступним кроком є визначення оптимальні системи збуту відповідно до закупівельної поведінки різних категорій цільових клієнтів.

Таблиця 7.22 – Формування системи збуту

| *№ п/п* | *Специфіка закупівельної поведінки цільових клієнтів* | *Функції збуту, які має виконувати постачальник товару* | *Глибина каналу збуту* | *Оптимальна система збуту* |
|---|---|---|---|---|
| 1 | Індивідуальні клієнти та представники невеликих підприємств, які заходять на сайт | Інформування клієнтів про функціональність, ціни та умови на сайті | Канал нульового рівня | Надання послуг через інтерфейс на сайті, біллінг через особистий кабінет користувача (підприємства) на сайті |
| 2 | Більші підприємства укладають угоди після розгляду пропозицій | Складання пропозицій, спілкування з замовниками, виконання тестових завдань і демонстрацій | Канал нульового рівня | |
| 3 | Підприємства купуватимуть проаналізовані дані в підприємства-партнера, яке їх створює та аналізує за допомогою нашого продукту | Створення повного ланцюжка продукування проаналізованих даних – від їх створення до обробки за допомогою нашого продукту | Канал одного рівня | Аналіз даних, що постачає підприємство-партнер, оплата згідно контракту |

Отже, в основному збут відбуватиметься напряму через сайт веб-сервісу. Для спеціалізованих рішень співробітники стартапу контактуватимуть із клієнтами для укладання угод і обговорення вимог. Іншим підходом є продаж послуг через партнера, який збиратиме дані та використовуватиме наш стартап як ланку у своєму процесі аналізу даних, результати якого партнер продаватиме своїм клієнтам.

Завершимо складання маркетингової програми визначенням підходів до маркетингових комунікацій відповідно до поведінки клієнтів із виділених цільових груп.

Таблиця 7.23 – Концепція маркетингових комунікацій

| *№ п/п* | *Специфіка поведінки цільових клієнтів* | *Канали комунікацій, якими користуються цільові клієнти* | *Ключові позиції, обрані для позиціонування* | *Завдання рекламного повідомлення* | *Концепція рекламного звернення* |
|---|---|---|---|---|---|
| 1 | Індивідуальні користувачі та невеликі підприємства орієнтуються на інтернет-позиціонування компанії | e-mail, чат на сайті | Продукт для широкої аудиторії, що вирішує широке коло задач | Показати переваги в простоті, інноваційності та якості над конкурентами | За допомогою інтернет-реклами, публікацій (whitepaper, у ЗМІ), на конференціях привернути увагу клієнтів і показати переваги продукту |
| 2 | Великі підприємства орієнтуються як на інтернет-позиціонування, так і на репутацію компанії, відповідність продукту різним стандартам | e-mail, телефон, особисте спілкування | Окрім продукту для широкої аудиторії – спеціалізовані рішення під конкретні задачі; моделі розгортання, які забезпечують data protection compliance | Показати інноваційність, стабільність і високу якість продукту, увагу до стандартів і специфічних вимог клієнтів | За допомогою тих же самих методів, що і в пункті 1 залучити потенційних клієнтів до більш детального спілкування з презентаціями та тестування ними продукту для демонстрації його переваг над конкурентами |

Таким чином, в основному будуть використовуватися електронні канали комунікації з клієнтами та розміщення реклами на ресурсах мережі Інтернет. Специфіка роботи з великими клієнтами вимагатиме зустрічей для демонстрації можливостей продукту та його переваг, а також обговорення найкращих підходів до задоволення їх специфічних потреб.

## 7.6 Висновки

Згідно проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів; хоча бар’єри входження є досить високими, є можливість їх подолати за допомогою пропозиції нових послуг і розробки інноваційного продукту.

Для успішного виконання проекту необхідно реалізувати сервіс із використанням нейронних мереж для класифікації зображень. В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

Потенційний успіх розроблюваного стартап-проекту обумовлюється тим, що сервіс працюватиме в галузі AI, які є однією з найбільш трендових у світі та стрімко зростає. Апаратне забезпечення, необхідне для інтелектуального аналізу даних, зараз знаходиться на достатньому рівні для успішного вирішення складних задач, які є в різних індустріях. Глобальний тренд на автоматизацію рутинної ручної праці, а також постійне здешевлення обчислювальних ресурсів дають гарне підґрунтя для економічного успіху стартапів.

Серед потенційних загроз було зазначено значну конкуренцію та велику кількість закритих наукових доробок, які використовуються сервісами-конкурентами. До того ж, конкуренти – це в основному спеціальні відділи великих IT-компаній, які мають кращий доступ до капіталу та кваліфікованих кадрів.

Мінімізувати ризик негативного вплив цих загроз пропонується за допомогою пропозиції нових продуктів, які мало представлені в конкуруючих компаній (таких як аналіз відео), а також надання технічної можливості розгортати код сервісу на потужностях замовників, що не пропонується конкурентами, але допоможе задовольнити найсуворіші вимоги до захисту даних від клієнтів, у яких є відповідно вимоги (конфіденційність медичних даних, приватність користувачів). При цьому інтелектуальна власність буде захищена як технічними, так і юридичними методами.

Також пропонується зважати на можливе сповільнення росту галузі AI, яка наразі знаходиться на стадії завищених сподівань. Стартап-проект провадитиме діяльність із більш консервативними економічними прогнозами, а при реалізації робитиметься фокус на ефективності розробленої системи для мінімізації собівартості виконання обчислень.

Окрім планів щодо початкового запуску стартапу у вигляді сервісу, що здатен задовольняти широке коло потреб клієнтів із різноманітними задачами, окреслено

також подальшу стратегію, коли користувачам пропонуватиметься розробка спеціалізованих рішень «під ключ» командами інженерів і дослідників стартапу.

# ВИСНОВКИ

У даній роботі було проведено дослідження використання розподіленого глибинного навчання до інтелектуального аналізу відео на прикладі вирішення задачі класифікації людських дій.

Було досліджено теоретичні засади вирішення задачі класифікації відео засобами машинного навчання, які включають згорткові та рекурентні нейронні мережі. Також було досліджено методи передобробки звукових даних і обчислення оптичного потоку.

Під час аналізу існуючих рішень було досліджено багатопотокові архітектури, що враховують деякі або всі з цих потоків: просторовий (послідовність кадрів), часовий (рух, представлений оптичними потоками), звуковий (аудіодоріжка). Можна виділити два основні типи архітектур. Перший використовує двовимірні згорткові нейронні мережі, які обчислюють представлення кадрів, а послідовність цих представлень обробляється рекурентною мережею. Другий підхід не застосовує рекурентні нейромережі, натомість використовуються 3D згорткові мережі.

За результатами аналізу існуючих рішень було виокремлено деякі проблеми існуючих підходів. Одна із них пов’язана з виконанням класифікації окремих кадрів або сегментів відео з подальшою агрегацією результатів; це призводить до втрати довгочасного контексту та порядку слідування цих фрагментів, а також зменшує точність класифікації відеопослідовностей, у яких цільова дія займає лише невелику частину загальної тривалості кліпу. Ця проблема робить підходи, що базуються лише на використанні згорткових нейромереж (без моделей, що обробляють послідовності), менш придатними для довгих відео та розпізнавання складних дій. Інший недолік, який є в багатопотокових архітектурах, пов’язаний із агрегацію передбачень, виконаних на кожному потоці окремо; таким чином модель синтезу не враховує особливості окремих потоків, а лише об’єднує ймовірності, передбачені різними моделями. Також підходи, засновані на 3D CNN, є менш ефективними з точки зору використання ресурсів і тренувальних даних: через велику кількість

параметрів потрібно більше обчислювальних потужностей, а також більше різноманітних даних.

З урахуванням цього аналізу було запропоновано багатопотоковий метод, у якому поглиблюється використання ідеї вивчення представлень. Вхідні дані кожного потоку (кадри, оптичні потоки, сегменти аудіо) обробляються згортковою нейромережею, яка повертає вектори представлень; послідовність цих векторів представлень подається на вхід рекурентній нейромережі, яка, в свою чергу, повертає вектор представлення відео. Підсумкові результати класифікації обчислюються моделлю синтезу, яка на відміну від існуючих рішень приймає на вхід вектор представлення відео, що характеризує всі три потоки. Застосування векторів представлення є більш гнучким способом, що дозволяє використовувати різні моделі синтезу, а також застосовувати ці компактні представлення у інших моделях. Як приклад такої моделі, було запропоновано їх використання у автокодувальнику, за допомогою якого можна вирішити задачу виявлення аномалій.

Запропонований метод було розроблено та реалізовано з урахуванням сучасних загальноприйнятих підходів до розробки моделей машинного навчання. Використовуються нові архітектури для обчислення векторів представлення кадрів і сегментів аудіо, а також сучасний метод визначення оптичного потоку за допомогою згорткових нейромереж. Спрощення навчання розробленої моделі досягається за рахунок окремого тренування моделей для різних потоків, а також використання передавального навчання.

Також було досліджено проблеми змагальних прикладів та деякі засоби підвищення стійкості моделі до них, а також підвищення точності класифікації, за допомогою розширення даних та змагального тренування. Окрім цього, було запропоновано метод адаптивної вибірки, який використовує оптичний потік для варіювання частоти вибірки так, щоб сегменти відео з активнішим рухом представлялися в тренувальному наборі даних більшою кількістю тренувальних прикладів. Також було показано, що механізм уваги є корисним інструментом інтерпретації передбачень моделі. Було виявлено, що використання капсульних

нейромереж, які є потенційним шляхом подальшого вдосконалення рішення, наразі не є можливим для великих наборів даних через повільне навчання та низьку точність.

Зрештою, було досліджено практичні аспекти реалізації та навчання розробленого методу, а саме апаратне та програмне забезпечення, а також підходи до розподіленого навчання моделей.

Було проведено експеримент, у ході якого було визначено оптимальну архітектуру рекурентної нейронної мережі для цієї задачі. Також було експериментально доведено можливість розподіленого навчання розробленої моделі.

Насамкінець, було проведено повне навчання розробленої моделі та виміряно точність класифікації на обраному наборі відео (UCF101). Запропонований метод досягає точності 93.1%, що є найкращим результатом для архітектур, котрі базуються на двовимірних згорткових та рекурентних нейронних мережах, а також переважає тривимірні згорткові нейромережі, натреновані лише на цьому наборі даних. Таким чином, розроблений метод є ефективним, оскільки дозволяє успішно моделювати відносно невеликі набори даних, і при цьому досягає високого рівня точності; також результатом його роботи є вектори представлень відео, що можуть використовуватися іншими моделями для вирішення різноманітних задач.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Gradient-based learning applied to document recognition / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. // Proceedings of the IEEE. – 1998. – №86(11). – С. 2278–2324.
2. Krizhevsky A. ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton. // Advances in Neural Information Processing Systems (NIPS). – 2012. – №25. – С. 1097–1105.
3. Simonyan K. Very deep convolutional networks for large-scale image recognition / K. Simonyan, A. Zisserman. // CoRR. – 2014. – abs/1409.1556.
4. Going deeper with convolutions / [C. Szegedy, W. Liu, Y. Jia та ін.]. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – С. 1–9.
5. Deep residual learning for image recognition / K. He, X. Zhang, S. Ren, J. Sun. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – С. 770–778.
6. Sabour S. Dynamic routing between capsules / S. Sabour, N. Frosst, G. E. Hinton. // Advances in Neural Information Processing Systems (NIPS). – 2017. – С. 3859–3869.
7. Sabour S. Matrix capsules with EM routing / S. Sabour, N. Frosst, G. E. Hinton. // International Conference on Learning Representations (ICLR). – 2018.
8. Goodfellow I. J. Explaining and harnessing adversarial examples / I. J. Goodfellow, J. Shlens, C. Szegedy. // CoRR. – 2014. – abs/1412.6572.
9. Schmidhuber J. Long short-term memory / J. Schmidhuber, S. Hochreiter. // Neural Computation. – 1997. – №9(8). – С. 1735–1780.
10. Empirical evaluation of gated recurrent neural networks on sequence modeling / J. Chung, C. Gulcehre, K. Cho, Y. Bengio. // CoRR. – 2014. – abs/1412.3555.
11. Recurrent models of visual attention / V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu. // Advances in Neural Information Processing Systems (NIPS). – 2014. – С. 2204–2212.

12. CNN architectures for large-scale audio classification / [S. Hershey, S. Chaudhuri, D. P. Ellis та ін.]. // IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2017. – С. 131–135.
13. HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent / F. Niu, B. Recht, C. Re, S. J. Wright. // Advances in Neural Information Processing Systems (NIPS). – 2011. – С. 693–701.
14. Learning spatiotemporal features with 3D convolutional networks / [D. Tran, L. Bourdev, R. Fergus та ін.]. // ICCV '15 Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). – 2015. – С. 4489–4497.
15. FlowNet: Learning optical flow with convolutional networks / [A. Dosovitskiy, P. Fischer, E. Ilg та ін.]. // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – 2015. – С. 2758–2766.
16. Штучна нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа.
17. Cybenko G. Approximation by superposition of sigmoidal functions / George Cybenko. // Mathematics of Control, Signals and Systems. – 1989. – №2(4). – С. 303–314.
18. Linnainmaa S. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors / Seppo Linnainmaa. // Master's Thesis (in Finnish), Univ. Helsinki. – 1970.
19. Nesterov Y. A method of solving a convex programming problem with convergence rate O (1/k2) / Yurii Nesterov. // Soviet Mathematics Doklady. – 1983. – №27(2). – С. 372–376.
20. Kingma D. P. Adam: A method for stochastic optimization / D. P. Kingma, J. L. Ba. // CoRR. – 2014. – abs/1412.6980.
21. Згорткова нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.
22. Convolutional neural networks [Електронний ресурс] – Режим доступу до ресурсу: http://cs231n.github.io/convolutional-networks/.

23. Britz D. Recurrent neural networks tutorial [Електронний ресурс] / Denny Britz. – 2015. – Режим доступу до ресурсу: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.
24. Rumelhart D. E. Learning internal representations by error propagation / D. E. Rumelhart, G. E. Hinton, R. J. Williams // Parallel Distributed Processing: Explorations in the Microstructure of Cognition / D. E. Rumelhart, G. E. Hinton, R. J. Williams. – MA, USA: MIT Press Cambridge, 1985. – (1).
25. Christopher O. Understanding LSTM Networks [Електронний ресурс] / Olah Christopher. – 2015. – Режим доступу до ресурсу: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
26. Lucas B. D. An iterative image registration technique with an application to stereo vision / B. D. Lucas, T. Kanade. // IJCAI'81 Proceedings of the 7th international joint conference on Artificial intelligence. – 1981. – №2. – С. 674–679.
27. Cooley J. W. An algorithm for the machine calculation of complex Fourier series / J. W. Cooley, J. W. Tukey. // Mathematics of Computation. – 1965. – №19(90). – С. 297–301.
28. Gibson J. J. The perception of the visual world / James J. Gibson. – Oxford, England: Houghton Mifflin, 1950. – 235 с.
29. Simonyan K. Two-stream convolutional networks for action recognition in videos / K. Simonyan, A. Zisserman. // Advances in Neural Information Processing Systems (NIPS). – 2014. – С. 568–576.
30. Soomro K. UCF101: A dataset of 101 human actions classes from videos in the wild / K. Soomro, A. R. Zamir, M. Shah. // CoRR. – 2012. – abs/1212.0402.
31. ImageNet [Електронний ресурс] – Режим доступу до ресурсу: http://www.image-net.org/.
32. Multi-stream multi-class fusion of deep networks for video classification / [Z. Wu, Y. Jiang, X. Wang та ін.]. // Proceedings of the 2016 ACM on Multimedia Conference. – 2016. – С. 791–800.
33. Efficient estimation of word representations in vector space / T. Mikolov, K. Chen, G. Corrado, J. Dean. // CoRR. – 2013. – abs/1301.3781.

34. Large-scale video classification with convolutional neural networks / [A. Karpathy, G. Toderici, S. Shetty та ін.]. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2014. – С. 1725–1732.
35. Carreira J. Quo vadis, action recognition? A new model and the kinetics dataset / J. Carreira, A. Zisserman. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – С. 4724–4733.
36. The kinetics human action video dataset / [W. Kay, J. Carreira, K. Simonyan та ін.]. // CoRR. – 2017. – abs/1705.06950.
37. Hara K. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? / K. Hara, H. Kataoka, Y. Satoh. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2018.
38. Ranjan A. Optical flow estimation using a spatial pyramid network / A. Ranjan, M. J. Black. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017.
39. FlowNet 2.0: Evolution of optical flow estimation with deep networks / [E. Ilg, N. Mayer, T. Saikia та ін.]. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017.
40. MPI Sintel Flow Dataset [Електронний ресурс] – Режим доступу до ресурсу: http://sintel.is.tue.mpg.de/.
41. Hu S. Video2Vec: Learning semantic spatio-temporal embeddings for video representation / S. Hu, Y. Li, B. Li. // International Conference on Pattern Recognition (ICPR). – 2016. – №23. – С. 811–816.
42. Inception-v4, Inception-ResNet and the impact of residual connections on learning / C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi. // AAAI. – 2017. – №4.
43. Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift / C. Szegedy, S. Ioffe. // ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning. – 2015. – №37. – С. 448–456.

44. Dropout: A simple way to prevent neural networks from overfitting / [N. Srivastava, G. Hinton, A. Krizhevsky та ін.]. // Journal of Machine Learning Research. – 2014. – №15(1). – С. 1929–1958.
45. Long-term recurrent convolutional networks for visual recognition and description / [J. Donahue, L. H. Hendricks, M. Rohrbach та ін.]. // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2015. – С. 2625–2634.
46. Audio Set: An ontology and human-labeled dataset for audio events / [J. F. Gemmeke, D. P. Ellis, D. Freedman та ін.]. // IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2017. – С. 776–780.
47. Su J. One pixel attack for fooling deep neural networks / J. Su, D. V. Vargas, K. Sakurai. // CoRR. – 2017. – abs/1710.08864.
48. Attention is all you need / [A. Vaswani, N. Shazeer, N. Parmar та ін.]. // Advances in Neural Information Processing Systems (NIPS). – 2017. – С. 6000–6010.
49. Kingma D. P. Auto-encoding variational bayes / D. P. Kingma, M. Welling. // CoRR. – 2013. – abs/1312.6114.
50. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/.
51. PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/.
52. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/.
53. Keras [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/.
54. Dettmers T. How to Parallelize Deep Learning on GPUs Part 1/2: Data Parallelism [Електронний ресурс] / Tim Dettmers – Режим доступу до ресурсу: http://timdettmers.com/2014/10/09/deep-learning-data-parallelism/.
55. Dettmers T. How to Parallelize Deep Learning on GPUs Part 2/2: Model Parallelism [Електронний ресурс] / Tim Dettmers – Режим доступу до ресурсу: http://timdettmers.com/2014/11/09/model-parallelism-deep-learning/.
56. Krizhevsky A. One weird trick for parallelizing convolutional neural networks / Alex Krizhevsky. // CoRR. – abs/1404.5997.

57. The marginal value of adaptive gradient methods in machine learning / [A. C. Wilson, R. Roelofs, M. Stern та ін.]. // Advances in Neural Information Processing Systems (NIPS). – 2017. – С. 4151–4161.

# ДОДАТОК А

## Лістинг програми

### Рекурентна модель

```
from keras import Model, Input
from keras.layers import Dense, Dropout, Bidirectional, CuDNNGRU, Activation, Permute, Reshape, merge
from keras.optimizers import SGD
from keras.regularizers import l2

from utils import load_weights


class RNNModel:
    def __init__(self, weights_path, num_classes, seq_length, feature_length, use_attention=False,
use_svm=False):
        if use_attention:
            self.inputs, self.outputs = self.create_model_with_attention(seq_length, feature_length)
        else:
            self.inputs, self.outputs = self.create_model(seq_length, feature_length)

        self.num_classes = num_classes

        if use_svm:
            self.model = self.get_model_with_svm()
        else:
            self.model = self.get_model()

        load_weights(self.model, weights_path)

    @staticmethod
    def create_model_with_attention(seq_length, feature_length):
        inputs = Input(shape=(seq_length, feature_length))
        m = RNNModel.attention_3d_block(inputs, seq_length)
        m = Dropout(0.5)(m)
        m = Bidirectional(CuDNNGRU(512, return_sequences=True))(m)
        m = Dropout(0.5)(m)
        m = Bidirectional(CuDNNGRU(512))(m)
        m = Dropout(0.5)(m)
        return inputs, m

    @staticmethod
    def create_model(seq_length, feature_length):
        inputs = Input(shape=(seq_length, feature_length))
        m = Dropout(0.5, input_shape=(seq_length, feature_length))(inputs)
        m = Bidirectional(CuDNNGRU(512, return_sequences=True))(m)
        m = Dropout(0.5)(m)
        m = Bidirectional(CuDNNGRU(512))(m)
        m = Dropout(0.5)(m)
        return inputs, m

    @staticmethod
    def compile(model, loss):
        print("RNN model: ")
        model.summary()
        metrics = ['accuracy', 'top_k_categorical_accuracy']
        optimizer = SGD(momentum=0.9, nesterov=True)
        model.compile(loss=loss, optimizer=optimizer, metrics=metrics)
        return model

    def get_model_with_svm(self):
        o = Dense(self.num_classes, kernel_regularizer=l2(0.01))(self.outputs)
```

```
        self.outputs = Activation('linear')(o)
        model = Model(inputs=self.inputs, outputs=self.outputs)
        return self.compile(model, 'categorical_hinge')

    def get_model(self):
        self.outputs = Dense(self.num_classes, activation='softmax')(self.outputs)
        model = Model(inputs=self.inputs, outputs=self.outputs)
        return self.compile(model, 'categorical_crossentropy')

    def get_keras_model(self) -> Model:
        return self.model

    @staticmethod
    def attention_3d_block(inputs, seq_length):
        # inputs.shape = (batch_size, time_steps, input_dim)
        input_dim = int(inputs.shape[2])
        a = Permute((2, 1))(inputs)
        a = Reshape((input_dim, seq_length))(a)
        a = Dense(seq_length, activation='softmax')(a)
        a_probs = Permute((2, 1), name='attention_vec')(a)
        output_attention_mul = merge([inputs, a_probs], name='attention_mul', mode='mul')
        return output_attention_mul
```

Згорткова модель для обчислення представлень кадрів

```
from keras import Model
from keras.applications import InceptionResNetV2
from keras.applications.inception_resnet_v2 import preprocess_input
from keras.layers import Dense, GlobalAveragePooling2D

from feature_extractor import FeatureExtractor

DEFAULT_SIZE = (299, 299)


class InceptionResNetV2Model:
    def __init__(self, num_classes, weights_file=None):
        self.model = self.create_model(num_classes)
        if weights_file:
            print(f"Loading weights from {weights_file}")
            self.model.load_weights(weights_file)
        else:
            print("Using pre-trained ImageNet weights")

    @staticmethod
    def create_model(num_classes) -> Model:
        base_model = InceptionResNetV2(include_top=False)
        x = base_model.output
        x = GlobalAveragePooling2D(name="avg_pool")(x)
        predictions = Dense(num_classes, activation='softmax')(x)
        model = Model(inputs=base_model.input, outputs=predictions)
        model.summary()
        return model

    def create_extractor(self) -> FeatureExtractor:
        """Returns a model that outputs 1536-dimensional embedding vector with shape=(None, 1536)"""
        extractor_model = Model(inputs=self.model.input, outputs=self.model.get_layer('avg_pool').out-
put)
        print(f"CNN extractor model:")
        extractor_model.summary()
        extractor = FeatureExtractor(extractor_model, preprocessing_fn=preprocess_input, tar-
get_size=DEFAULT_SIZE)
        return extractor
```

Згорткова модель для оптичних потоків

```
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Dense, Flatten, Dropout
from keras.layers.normalization import BatchNormalization
from keras.models import Sequential
from keras.optimizers import Adam


class OpticalFlowCNNModel:
    def __init__(self, width, height, channels, output_classes):
        self.model = OpticalFlowCNNModel.create_model(width, height, channels, output_classes)

    @staticmethod
    def build_model(width, height, channels, output_classes):
        model = Sequential()

        model.add(Conv2D(96, kernel_size=(7, 7), strides=(2, 2), input_shape=(width, height, channels),
                         activation='relu', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

        model.add(Conv2D(256, kernel_size=(5, 5), strides=(2, 2), activation='relu', padding='same'))
        model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

        model.add(Conv2D(512, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'))
        model.add(Conv2D(512, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'))

        model.add(Conv2D(512, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'))
        model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same'))

        model.add(Flatten())
        model.add(Dense(4096, activation='relu'))
        model.add(Dropout(0.5))

        model.add(Dense(2048, activation='relu'))
        model.add(Dropout(0.5))

        if output_classes:
            model.add(Dense(output_classes, activation='softmax'))

        return model

    @staticmethod
    def compile_model(model):
        metrics = ['accuracy', 'top_k_categorical_accuracy']
        optimizer = Adam()
        model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=metrics)
        return model

    @staticmethod
    def create_model(width, height, channels, output_classes):
        model = OpticalFlowCNNModel.build_model(width, height, channels, output_classes)
        return OpticalFlowCNNModel.compile_model(model)

    def get_keras_model(self):
        return self.model
```

Тренування рекурентної моделі просторового потоку

```
import numpy

from keras.callbacks import TensorBoard, ModelCheckpoint, EarlyStopping, CSVLogger, ReduceLROnPlateau
from data import DataSet
import time
import os.path

from rnn import RNNModel
from utils import parse_weights_path_from_args

timestamp = time.time()

model_name = 'lstm-' + str(int(timestamp))
batch_size = 32
epochs = 1000
data_type = 'features'
seq_length = 180
num_classes = 101
feature_length = 1536


def calculate_steps_per_epoch(data, train_test, batch_size):
    samples = data.get_num_samples(train_test)
    return samples // batch_size


def train(weights_path):
    checkpointer = ModelCheckpoint(
        filepath=os.path.join('data', 'checkpoints', model_name + '-' + '.{epoch:03d}-
{val_loss:.3f}.hdf5'),
        verbose=1,
        save_best_only=True,
        save_weights_only=True)
    data = DataSet()

    tb = TensorBoard(log_dir=os.path.join('data', 'logs', model_name))

    early_stopper = EarlyStopping(patience=10)

    csv_logger = CSVLogger(
        os.path.join('data', 'logs', 'experiment', model_name + '-' + 'training-' + str(timestamp) +
'.log'))

    reduce_lr = ReduceLROnPlateau(monitor='val_loss', min_lr=1e-6, patience=5, verbose=1)
    train_steps_per_epoch = calculate_steps_per_epoch(data, 'train', batch_size)

    generator = data.sample_generator(batch_size, seq_length, 'train', 'features')
    val_generator = data.sample_generator(batch_size, seq_length, 'test', 'features')

    model = RNNModel(weights_path, num_classes, seq_length, feature_length).get_keras_model()


    model.fit_generator(
        generator=generator,
        steps_per_epoch=train_steps_per_epoch,
        epochs=epochs,
        verbose=1,
        callbacks=[tb, csv_logger, reduce_lr, early_stopper, checkpointer],
        validation_data=val_generator,
        validation_steps=calculate_steps_per_epoch(data, 'test', batch_size),
        workers=1)


if __name__ == '__main__':
    train(parse_weights_path_from_args())
```